RL-TR-91-190
Final Technical Report
August 1991

REASONING WITH INCOMPLETE AND UNCERTAIN INFORMATION

General Electric Company

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. 5305

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-91-190 has been reviewed and is approved for publication.

APPROVED: Chuian-Chuian Hwong

CHUIAN-CHUIAN HWONG
Project Engineer

FOR THE COMMANDER: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
COMMAND, CONTROL & COMMUNICATIONS DIRECTORATE

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL(C3CA) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REASONING WITH INCOMPLETE AND UNCERTAIN INFORMATION

Piero P. Bonissone
Jonathan P. Stillman
James K. Aragone
Nancy C. Wood

Contractor: General Electric Company
Contract Number: F30602-85-C-0033
Effective Date of Contract: 14 January 1985
Contract Expiration Date: 15 April 1991
Short Title of Work: Reasoning with Incomplete and Uncertain Information

Period of Work Covered: May 87 - Aug 90

Principal Investigator: Piero P. Bonissone
Phone: (518) 387-5155

RL Project Engineer: Chuian-Chuian Hwong
Phone: (315) 330-4833

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Chuian-Chuian Hwong, RL (C3CA), Griffiss AFB NY 13441-5700 under Contract F30602-85-C-0033.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Final May 87 - Aug 90

4. TITLE AND SUBTITLE
REASONING WITH INCOMPLETE AND UNCERTAIN INFORMATION

5. FUNDING NUMBERS
C - F30602-85-C-0033
PE - 62301E
PR - E305
TA - 00
WU - 01

6. AUTHOR(S)
Piero P. Bonissone, Jonathan P. Stillman, James K. Aragone, Nancy C. Wood

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
General Electric Company
Corporate Research & Development
P.O. Box 8
Schenectady NY 12301

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington VA 22209

Rome Laboratory (C3CA)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-91-190

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Chuian-Chuian Howong/C3CA/(315) 330-4833

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report summarizes research efforts in the area of Reasoning with Imcomplete and Uncertain Information. The progress reported in this paper is mainly in three areas. The first is in applying RUM (Reasoning With Uncertainty Module) to different applications. The second is in developing the semantics of nonmonotonic reasoning. The third is in developing methods for combining nonmonotonic reasoning with uncertainty. This report consists of ten chapters, which have been, in part or full, previously published as papers in technical meetings or professional journals. These papers describe the theoretical advances that culminated with the development and application of PRIMO (Plausible Reasoning Module), a software tool implemented in Common Lisp and Flavors at GE. PRIMO is a reasoning system that integrates the theories of plausible and defaults reasoning. It consists of a language for representing uncertain and default knowledge, along with algorithms for reasoning in this language. PRIMO handles uncertain information by qualifying each possible value assignment to any given variable with an uncertainty interval. PRIMO handles imcomplete information by evaluation nonmonotonic justified (NMJ) rules.

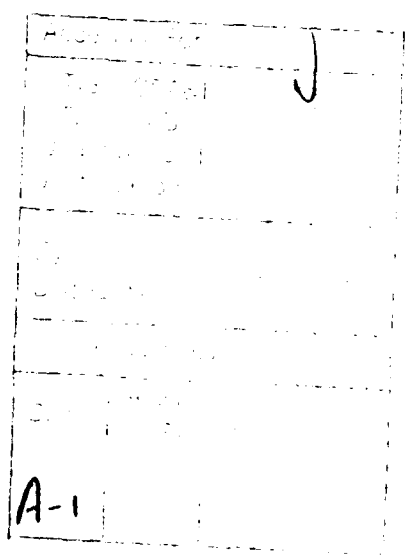
14. SUBJECT TERMS Expert Systems, Artificial Intelligence, Inference Mechanisms, Knowledge Representation, Reasoning with Uncertainty, Default Reasoning

15. NUMBER OF PAGES 230

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

Contents



Introduction

This report summarizes our research efforts in the area of Reasoning with Incomplete and Uncertain Information.

This report consists of ten chapters, which have been, in part or full, previously published as papers in technical meetings, and professional journals. These papers describe the current theoretical and technological advances that will culminate with the development and application of PRIMO (Plausible ReasonIng MOdule), a software tool implemented in Common Lisp and Flavors at GE.

PRIMO is a resoning system that integrates the theories of plausible and default reasoning. It consists of a language for representing uncertain and default knowledge, along with algorithms for reasonming in this language.

PRIMO handles uncertain information by qualifying each possible value assignment to any given variable with an uncertainty interval. The interval's lower bound represents the minimal degree of confirmation for the value assignment. The upper bound represents the degree to which the evidence failed to refute the value assignment. The interval's length represents the amount of ignorance attached to the value assignment. The uncertainty intervals are propagated and aggregated by Triangular norm based uncertainty calculi.

PRIMO handles incomplete information by evaluating non-monotonic justified (NMJ) rules. These rules express the knowledge engineer's preference or bias to be used by the reasoning system in cases of total or partial ignorance regarding the value assignment of a given variable. The NMJ rules are used when there is no plausible evidence (to a given numerical standard of belief or certainty) to infer that a given value assignment is either true or false. The conclusions of NMJ rules can be retracted by the belief revision system, when enough plausible evidence is available.

For efficiency considerations restrictions are placed on the types of rules allowed in PRIMO. The monotonic rules are non-cyclic Horn clauses, and are maintained by a linear belief revision algorithm operating on a rule graph. The NMJ rules can have cycles, but cannot have disjunctions in their conclusions. By identifying sets of NMJ rules as strongly connected components (SCC's), we can decompose the rule graph into a directed acyclic graph (DAG) of nodes, some of which are SCCs with several input edges and output edges. PRIMO contains algorithms to efficiently propagate uncertain and incomplete information through the above structures at run time. These algorithms require finding satisfying assignments for nodes in each SCC, and are thus NP hard in

the unrestricted case. By restricting the size and complexity of the SCC's, precomputing their structural information, and using run-time evaluated certainty measures to break the symmetry of multiple fixed points, we can achieve tractability in the average case. The semantics and algorithms used in PRIMO are described in more detail in chapter 6.

The papers taken as a group represent the progress of our work over the past 18 months. They are in some sense prototypical of the evolution of PRIMO. We now give a thorough summary of the papers collected in this report.

The first paper, "A Study on Uncertainty Management" is a report on the state of the art of reasoning with uncertainty. This study analyzes the various sources of uncertainty, the state of the art of reasoning theories and technologies capable of dealing with uncertainty, their applicability to Advanced Crew Station Programs, such as the Submarine Operational Automation System (SOAS) program, and their computational cost.

The second paper, "T-norm Based Reasoning in Situation Assessment Applications," is a report on the use of RUM to perform Situation Assessment (SA). The paper consists of a summary of RUM and T-norm based reasoning, a sequence of experiments, and a description of the test-bed environment for developing these experiments. The sequence of experiments in both naval and aerial SA consisted of correlating reports and tracks, locating and classifying platforms, and identifying intents and threats. The paper illustrates an example of naval SA. The test-bed environment has been provided for by LOTTA, a symbolic simulator implemented in Flavors. This simulator maintains time-varying situations in a multi-player antagonistic game where players must make decisions in light of uncertain and incomplete data. RUM has been used to assist one of the LOTTA players to perform the SA task.

While the second paper deals mainly with RUM and T-norm based reasoning, the third paper, "Plausible Reasoning in Dynamic Classification Problems," deals mainly with the test-bed architecture and a methodology for testing and validating the knowledge base and inference techniques used for dynamic classification problems. The test-bed architecture is composed of two parts: a simulation environment, LOTTA, and a reasoning system, RUM.

The simulation environment is composed of four basic modules: *the window subsystem*, a window based user interface for displaying time varying features; *LOTTA*, the simulator; and a set of tools for interfacing to a reasoning system. LOTTA has no reasoning capabilities; these are provided by external reasoning modules, easily interfaced to the LOTTA data structures.

RUM and RUMrunner, RUM's run-time counterpart, are the reasoning systems used in the test-bed architecture. RUM's main function is to build rule-based reasoning systems following the rapid prototyping methodology. Following the testing, and verification of the application using RUM, the knowledge base is then automatically translated and compiled into compact data structures. RUMrunner reasons opportunistically with these data structures to achieve run-time performance required by most real-time applications.

The paper also reports on the use of this architecture in both naval and aerial SA problems. The architecture described in this paper has also been used for the testing, and

development of applications using PRIMO.

In earlier reports we presented a semantical account of nonmonotonic reasoning based on the partial ordering of interpretations of standard logics. The fourth paper, "New Results on Semantical Nonmonotonic Reasoning," generalizes and extends the earlier work. The paper elucidates the structural relation between the new work and the old. Also, in the paper the new results are applied to give a logical account of justification based truth maintenance.

The fifth paper, "Logics of Justified Belief," gives a formal semantics to truth maintenance by offering a mathematical logic - equipped with an underlying model theory - that is used to characterize quite precisely some well known models of truth maintenance. In addition to giving meaning to truth maintenance in terms of a formal logic, the paper shows that each characterising logic corresponds to a particular truth maintenance system and *vice versa*.

The sixth paper, "Uncertainty and Incompleteness: Breaking the Symmetry of Defeasible Reasoning," addresses two major difficulties in default logics, namely their intractability and the problem of selecting among multiple extensions. This paper proposes an apporoach to these problems based on integrating nonmonotonic reasoning with plausible reasoning based on triangular norms. The paper shows how RUM, which performs uncertain monotonic inferences on an acyclic graph, has been extended to allow nonmonotonic inferences and cycles within nonmonotonic rules. By restricting the size and *complexity* of the nonmonotonic cycles it can still perform efficient inferences. The uncertainty measures in RUM provide a basis for deciding between multiple defaults. Different algorithms and heuristics for finding the optimal defaults are discussed.

The seventh paper, "The Complexity of Horn Theories with Normal Unary Defaults", proves that although fast algorithms exist for determining whether a literal holds in a propositional default theory in which the propositional theory consists solely of literals and the default rules are *Horn*, and exist for deciding satisfiability of propositional Horn theories, the two cannot be combined without introducing intractability. In particular, we show that when the propositional theory of a default theory allows Horn clauses, the membership problem becomes intractable even when the default rules in the theory are restricted to being propositional *normal unary* default rules, a strong restriction of propositional Horn default rules. The paper also presents several related results, showing that the entailment problem, the enumeration problem, and the problem of determining whether there exists an extension that "satisfies" some specified number of the default rules are all intractable for these restricted default theories.

The eighth paper "It's Not My Default: The Complexity of Membership Problems in Restricted Propositional Default Logics," introduces a hierarchy of classes of propositional default rules, and characterizes the complexity of typical problems in those classes under various assumptions about the underlying propositional theory.

The ninth paper, "PRIMO: A Tool for Reasoning with Incomplete and Uncertain Information" reviews the theoretical foundations of PRIMO and discusses PRIMO's design and implementation.

The final paper, "PRIMO: User's Guide," brings together the work previously discussed. The paper describes the final implementation of PRIMO and the steps involved in developing an application.

1. A Study on Uncertainty Management

Piero P. Bonissone
General Electric Corporate Research and Development
Schenectady, New York 12301

1.1 Introduction

Uncertainty is a pervasive phenomenon throughout many environments. Consider, for example, the submarine environment. Organic and non-organic sensor inputs provide imprecise and, occasionally, unreliable or inaccurate data. The fusion of multi-sensor tracks into a consolidated contact track is marred with ambiguity caused by tracks crossing or by lost-and-recovered tracks. The knowledge used to define and interpret a given situation (scene) is often incomplete and imprecise, since it is usually based on subjective evaluations of similar situations. For a given situation, the matching of a tactical plan developed for some similar contingent situations is also an approximate process. Once a plan is selected, uncertainty is still present in the plan adaptation, projection and repair phases. Finally, during plan execution, we cannot deterministically predict the results of the performed actions.

This study analyzes the various sources of uncertainty, the state of the art of reasoning theories and technologies capable of dealing with uncertainty, their applicability to a domain of problems referred to as *the dynamic classification problem*, and their computational cost.

1.1.1 Uncertainty Sources

In a survey of reasoning with uncertainty [BT85], it is noted that there are two major types of uncertainty: *randomness* and *fuzziness*. Randomness deals with the uncertainty of whether a given element belongs or does not belong to a well-defined set (event). Fuzziness deals with the uncertainty derived from the partial membership of a given element to a set whose boundaries are not sharply defined.

These two types of uncertainty can be introduced in reasoning systems is caused by a variety of sources: the *reliability* of the information, the inherent *imprecision* of the representation language in which the information is conveyed, the *incompleteness* of the information, and the *aggregation* or summarization of information from multiple sources.

The first source type is related to the *reliability of information:* uncertainty can be present in the factual knowledge (i.e., the set of assertions or facts) due to inaccuracy and poor reliability of the instruments used to make the observations. Uncertainty can also

occur in the knowledge base (i.e., the rule set) as a result of using weak implications. Unlike categorical rules (describing set subsumption relationships) weak implications or plausible rules are typically used to describe likely interpretations of situations. By their very nature, these rules are less reliable than categorical rules and are used when the expert or model builder is unable to establish an exact correlation between premise and conclusion. In most expert systems the degree of implication is artificially expressed as a scalar value on an interval (certainty factor, conditional probability, degree of sufficiency, etc.). This value represents the change from the strict implication *for all* x, $A(x) \rightarrow B(x)$, to the weaker statement *for most* x, or *usually, for all* x, $A(x) \rightarrow B(x)$. The latter statement is not categorical and allows the possibility of exceptions to the rule. Thus the logical implication has now been changed into a plausible implication or *disposition* [Zad85b], [Zad88]. A natural way to express such a degree of implication is achieved by using fuzzy quantifiers such as *most, almost all*, etc. [Zad83a], [Zad84a]. [1]

Uncertainty in the data can be compounded by aggregating uncertain data in the premise, by propagating certainty measures to the conclusion, and by consolidating the final certainty measure of conclusions derived from different rules. Triangular norms and conorms [6], [DP84] can be used to generalize the conjunction and disjunction operators that provide the required aggregation capabilities. A description of their characteristics is provided in reference [1].

The second type of uncertainty is caused by the *inherent imprecision* of the facts and rules representation language. Observations can contain ill-defined concepts. Rules can contain vague predicates describing tests which cannot be expressed by boolean expressions (e.g., a great change in heading). As a result, these rules cannot be interpreted exactly. This problem has been partially addressed by the possibilistic theory of approximate reasoning that, in light of imprecise fact and rule descriptions, allows one to make weaker inferences based on a *generalized modus ponens* [Zad75].

The third type of uncertainty is caused by the *incompleteness* of the information. This type of uncertainty has generally been modeled by non-numerical characterizations, such as Doyle's Reasoned Assumptions [Doy83].

The fourth type of uncertainty arises from the *aggregation of information* from different knowledge sources or experts. When unconditional statements (facts) are aggregated, three potential problems can occur: the closure of the representation may no longer be preserved when the facts to be aggregated have *different granularity* (the single-valued certainty measures of the facts may change into an interval-valued certainty measure of the aggregated fact); the aggregation of conflicting statements may generate a *contradiction* that should be detected; the rule of evidence combination may create an over-estimated certainty measure of the aggregated fact, if a normalization is used to eliminate or *hide* a contradiction [Zad84b], [Zad85a]. The first two problems are typical of single-valued numerical approaches, while the last problem is found in the two-valued approach proposed by Dempster [Dem67].

[1] A fuzzy quantifier is a fuzzy number representing the relative cardinality of the subset of elements in the universe of discourse that *usually* satisfy the given property, i.e., the implication.

1.1.2 Focus and Structure of This Study

We have observed that there are different types and sources of uncertainty, and, correspondingly, there are different approaches for handling it. Each approach has its own underlying assumptions and semantics, as each approach captures different aspects of the uncertainty.

In this study will focus on the analysis of the trade-off between the adequacy of an approach and its computational cost. This analysis is motivated by the desire to meet two important requirements: the *scalability* of the Knowledge Base and the *functional extensibility* of the supporting architecture. To meet these requirements we will analyze the computational complexity, the input information requirements, the underlying assumptions and associated problem decomposition techniques needed to provide modularity, and the available approximations of each major approach.

In the next section (Section 1.2) we will review the state of the art of techniques for reasoning with uncertainty. We will emphasize the numerical approaches and contrast and compare probabilistic and possibilistic methods. These methods will be described in Section 1.3 and 1.4,

Section 1.5 will cover a subset of reasoning technologies embodying possibilistic theories. These theories and technologies are evaluated and compared against a list of requirements in Section 1.6.

In section 1.7, we describe some of the most relevant tasks in situation assessment and tactical planning. Finally, in Section 1.8, we discuss the applicability of uncertainty management techniques to these tasks.

1.2 State of the Art of Reasoning with Uncertainty

The existing approaches to representing uncertainty can be subdivided into two basic categories according to their *quantitative* or *qualitative* characterizations of uncertainty. (See references [2], [Pea88] for a survey). Among the quantitative approaches, we find two types of reasoning that differ in the semantics of their numerical representation. One is the *probabilistic reasoning* approach, based on probability theory. The other one is the *possibilistic reasoning* approach, based on the semantics of many-valued logics.

Some of the more traditional techniques found among the approaches derived from probability are based on *single-valued* representations. These techniques include Bayes Rule [Pea82, Pea85, Pea88], Modified Bayes Rule [DHN76] and Confirmation Theory [SB75]. A more recent trend among the probabilistic approaches is represented by approaches based on *interval-valued* representations such as Dempster-Shafer Theory [Dem67, Sha76], Evidential Reasoning [LGS86], and Probability Bounds, i.e., consistency and plausibility (see [Qui83]).

Over the last five years, considerable efforts have been devoted to improve the computational efficiency of Bayesian Belief Networks (BBN) for trees and small polytrees [Pea88a], and for directed acyclic graphs (influence diagrams) [HM84, Sch86, AR87].

Problem decomposition techniques (e.g. loopcuts, cliques) [LD88] and approximate methods (e.g. conditioning, clustering, bounding interval, simulations) [Hen87] have been derived to handle multi-connected Bayesian Belief Networks [Pea88a].

Among the approaches anchored on many-valued logics, the most notable are based on a *fuzzy-valued* representation of uncertainty. These include Necessity and Possibility Theory [Zad78, Zad79a], the Linguistic Variable Approach [Zad79b, Zad83b], and the Triangular-norm based approach [3, 1, 5, Bon89].

With numerical representations, it is possible to define a *calculus* that provides a mechanism for propagating uncertainty through the reasoning process. Similarly, the use of aggregation operators provides summaries which can then be ranked to perform rational decisions.

Models based on *qualitative* approaches, on the other hand, are usually designed to handle the aspect of uncertainty derived from the *incompleteness* of the information, such as Reasoned Assumptions [Doy83], and Default Reasoning [Rei80]. With a few exceptions, they are generally inadequate to handle the case of *imprecise* information, as they lack any measure to quantify confidence levels [Doy83]. A few approaches in this group have addressed the representation of uncertainty, using either a *formal* representation, such as Knowledge and Belief [YM86], or a *heuristic* representation, such as the Theory of Endorsements [Coh85, CG83].

The formal approach has a corresponding (modal) logic theory that determines the mechanism by which inferences (theorems) can be *proven* or *believed* to be true. The heuristic approach has a set of context-dependent rules to define the way by which frame-like structures (endorsements) can be combined, added or removed.

We will now focus our discussion on the two types of quantitative representations of uncertainty and we will contrast probabilistic and possibilistic reasoning systems.

1.2.1 Approximate Reasoning Systems

The task of a reasoning system is to determine the *truth value* of statements describing the state or the behavior of a real world system. However, this hypothesis evaluation requires complete and certain information, which is typically not available. Therefore, approximate reasoning techniques are used to determine a *set of possible worlds* that are logically consistent with the available information. These possible worlds are characterized by a set of propositional variables and their associated values. As it is generally impractical to describe these possible worlds to an acceptable level of detail, approximate reasoning techniques seek to determine some properties of the set of possible solutions or some constraints on the values of such properties [Rus87], [Rus89], [Rus90b].

A large number of approximate reasoning techniques have been developed over the past decade to provide these solutions. (See references [2], [Pea88] for a survey). These techniques have been roughly subdivided into two basic categories according to their *quantitative* or *qualitative* characterizations of uncertainty. Among the quantitative approaches, we find two types of reasoning that differ in the semantics of their numerical representation. One is the *probabilistic reasoning* approach, based on probability theory.

The other one is the *possibilistic reasoning* approach, based on the semantics of many-valued logics. We will briefly contrast these two types of quantitative representations and focus our discussion on possibilistic reasoning systems.

1.2.2 Probabilistic and Possibilistic Reasoning Systems

Probabilistic Reasoning Systems

Probability-based reasoning, or *probabilistic reasoning* seeks to describe the constraints on the variables that characterize the possible worlds with conditional probability distributions based on the evidence in hand. Their supporting formalisms are based on the concept of *set-measures*, additive real functions defined over certain subsets of some space.

These methods focus on chance of occurrence and relative likelihood. They are oriented primarily toward the choice of decisions that are optimal in the *long-run*, as they measure the *tendency* or *propensity* of truth of a proposition without assuring its actual validity. Thus, probabilistic reasoning estimates the frequency of the truth of a hypothesis as determined by prior observation (objectivist interpretation) or a degree of gamble based on the actual truth of the hypothesis (subjectivist interpretation).

Probabilistic methods seldom make categorical assertions about the actual state of the system being investigated. Rather, they indicate that there is an experimentally-determined (or believed) tendency or propensity for the system to be in some specified state.

The typical standard of measurement of probabilistic decision-making is, correspondingly, a measure of average decision utility that is meaningful only when the methodology is to be applied in a large number of situations. Probabilistic methods have a well-developed set of decision-theoretic approaches based primarily on the concept of expected utility [LR57]. Experience obtained through psychological experimentation [KST82] suggests, on the other hand, that human beings often misunderstand and misapply probabilistic information, thus reducing its potential value.

From a practical computational viewpoint, probabilistic methods suffer from problems associated with the reliable determination of all required joint and conditional probabilities. In complex systems, it is often the case that many variables interrelate with each other in ways that are not expressible in terms of simpler interactions. In a military assessment problem, for example, such quantities as "the probability of frontal attack given this situation" are not easily measured or elicited.

Possibilistic Reasoning Systems

Possibilistic reasoning, which is rooted in fuzzy set theory [Zad65] and many-valued logics, seeks to describe the constraints on the possible worlds in terms of their *similarity* to other sets of possible worlds.

These methods focus on *single* situations and cases. Rather than measuring the tendency of the given proposition to be valid, they seek to find another proposition that is valid. This proposition is usually less specific and resembles (according to some measure of similarity) the original hypothesis of interest.

Given the purpose and characteristics of probabilistic and possibilistic reasoning, it is clear that these technologies ought to be regarded as being complementary rather than competitive.

The single-case orientation of possibilistic techniques makes them particularly suitable for case-based reasoning. In CBR, it is typically the case that the problem in hand (probe) has never been encountered before. The inference in CBR is based on the existence of cases *similar enough* (i.e. close enough) to the probe to justify the adaptability of their solution to the current problem. The possibilistic techniques are also very suitable to represent the subjective degrees of belief inherent in the knowledge bases used to interpret and understand tactical situations. Typically these situations have never been encountered before, but the problem domain experts can describe and interpret similar, more generic, prototypical situations.

The notion of similarity is based on the concept of *metric* or distance, as opposed to that of set measure. Distances are functions which assign a number greater that zero to pairs of elements of some set (for sake of simplicity, we will assume the range of this function to be the interval [0,1]). Distances are *reflexive*, *commutative*, and *transitive*. Similarity can be defined as the complement of distance, i.e.:

$$S(A, B) = 1 - d(A, B)$$

The basic structural characteristics of the similarity functions is an extended notion of transitivity that allows the computation of bounds on the similarity between two objects A and B on the basis of knowledge of their similarities to a third object C:

$$S(A, B) \geq T(S(A, C), S(B, C)),$$

where T is a Triangular-norm [1], [3]. Any continuous triangular norm $T(A, B)$ falls in the interval $Max(0, A + B - 1) \leq T(A, B) \leq Min(A, B)$. Thus, we can observe that the if we use the lower bound of the range of T-norms in the expression describing the transitivity of similarity, we obtain the triangular inequality for distances. If we use the upper bound, we obtain the ultrametric inequality.

This similarity notion is a direct extension of the notion of *accessibility relation* that is of fundamental importance in modal logics. This notion is further described by Ruspini in reference [Rus90a]. In summarizing Ruspini's results, we can observe that the notion of accessibility captures the idea that whatever is true in some world w, is true, but in a modified sense, in another w' that is accessible from it. When considering multiple levels of accessibility (indexed by a number between 0 and 1), this relation, measuring the resemblance between two worlds, may be used to express the extent by which considerations applicable in one world may be extended to another world.

The basic inferential mechanism, underlying the *generalized modus-ponens* [Zad79b], makes use of inferential chains and the properties of a similarity function to relate the state of affairs in the two worlds that are at the extremes of an inferential chain.

1.3 Probabilistic Reasoning: Theories and Approaches

Having contrasted the difference between probabilistic and possibilistic reasoning techniques, we will now examine selected representative approaches. Among the probabilistic techniques we will analyze the Bayesian approaches (Bayesian, Modified Bayesian, Bayesian Belief Networks), Confirmation Theory (certainty factors) and Dempster-Shafer (Belief Theory).

1.3.1 Bayes Rule

Given a set of hypotheses $\mathbf{H} = \{h_1, h_2, \ldots, h_n\}$ and a sequence of pieces of evidence $\{e_1, e_2, \ldots, e_m\}$, Bayes rule, derived from the formula of conditional probability, states that the posterior probability $P(h_i \mid e_1, e_2, \ldots, e_m)$ can be derived as a function of the conditional probabilities $P(e_1, e_2, \ldots, e_m \mid h_i)$ and the prior probability $P(h_i)$:

$$P(h_i \mid e_1, e_2, \ldots, e_m) = \frac{P(e_1, e_2, \ldots, e_m \mid h_i) \cdot P(h_i)}{\sum_{i=1}^{n} P(e_1, e_2, \ldots, e_m \mid h_i) \cdot P(h_i)} \tag{1.1}$$

The Bayesian approach is based on two fundamental assumptions:

- Each hypothesis h_i is mutually exclusive with any other hypothesis in the set **H** and the set of hypotheses **H** is exhaustive, i.e.:

$$P(h_i, h_j) = 0 \text{ for } i \neq j \tag{1.2}$$

$$\sum_{i=1}^{n} P(h_i) = 1 \tag{1.3}$$

- Each piece of evidence e_j is conditionally independent under each hypothesis, i.e.:

$$P(e_1, e_2, \ldots, e_m \mid h_i) = \prod_{j=1}^{m} P(e_j \mid h_i) \tag{1.4}$$

Note that assumptions 1.2 and 1.3 are required to derive Bayes Rule from the formula of conditional probability. Assumption 1.4, on the other hand, is usually made to alleviate the difficulty of determining the conditional joint probability required by equation 1.1. Thus, under assumption 1.4, equation 1.1 becomes computationally feasible.

This method requires a large amount of data to determine the estimates for the prior and conditional probabilities. Such a requirement becomes manageable when the problem

can be represented as a *sparse* Bayesian network that is formed by a hierarchy of small cluster of nodes. In this case the dependencies among variables (nodes in the network) are known and only the explicitly required conditional probabilities must be obtained [Pea85].

1.3.2 Modified Bayes Rule

In addition to assumptions 1.2 and 1.3 (for derivational needs) and assumption 1.4 (for operational convenience) needed by the original Bayes Rule, the Modified Bayesian approach, used in PROSPECTOR, also requires that each piece of evidence e_j be conditionally independent under the *negation* of each hypothesis, i.e.:

$$P(e_1, e_2, \ldots, e_m \mid h_i) = \prod_{j=1}^{m} P(e_j \mid \neg h_i) \tag{1.5}$$

The Modified Bayesian approach is based on a variation of the odds-likelihood formulation of Bayes rule. When all the pieces of evidence are *certainly true*, this formulation defines the posterior odds as:

$$\begin{aligned} O(h_i \mid e_1, e_2, \ldots, e_m) &= \frac{P(e_1 \mid h_i)}{P(e_1 \mid \neg h_i)} \frac{P(e_2 \mid h_i)}{P(e_2 \mid \neg h_i)} \cdots \frac{P(e_n \mid h_i)}{P(e_n \mid \neg h_i)} \frac{P(h_i)}{P(\neg h_i)} \\ &= \lambda_{1,i}\lambda_{2,i} \cdots \lambda_{n,i} O(h_i) \end{aligned} \tag{1.6}$$

where:

$\lambda j, i = \frac{P(e_j \mid h_i)}{P(e_j \mid \neg h_i)}$ is the *likelihood ratio* of e_j for hypothesis h_i.

$O(h_i) = \frac{P(h_i)}{P(\neg h_i)}$ is the *odds* on hypothesis h_i.

An analogous odds-likelihood formulation is derived for the case when all the pieces of evidence are *certainly false*:

$$\begin{aligned} O(h_i \mid \neg e_1, \neg e_2, \ldots, \neg e_m) &= \frac{P(\neg e_1 \mid h_i)}{P(\neg e_1 \mid \neg h_i)} \cdot \frac{P(\neg e_2 \mid h_i)}{P(\neg e_2 \mid \neg h_i)} \cdots \frac{P(\neg e_n \mid h_i)}{P(\neg e_n \mid \neg h_i)} \cdot \frac{P(h_i)}{P(\neg h_i)} \\ &= \lambda^*_{1,i} \cdot \lambda^*_{2,i} \cdots \lambda^*_{n,i} \cdot O(h_i) \end{aligned} \tag{1.7}$$

The likelihood ratio $\lambda j, i$ measures the *sufficiency* of a piece of evidence e_j to prove hypothesis h_i. Similarly, $\lambda^*_{j,i}$ measures the *necessity* of such a piece of evidence to prove the given hypothesis (12).

Formulae 1.6 and 1.7 assume that evidence e_j is precise (i.e., $P(e_j) \in \{0, 1\}$). This is not the case in most expert system applications. Therefore, the above formulae must be modified to accommodate uncertain evidence. This is accomplished by using a linear interpolation formula. For the case of single evidence, the posterior probability $P(h_i \mid e'_j)$ is computed as:

$$P(h_i \mid e'_j) = P(h_i \mid e_j) \cdot P(e_j \mid e'_j) + P(h_i \mid \neg e_j) \cdot P(\neg e_j \mid e'_j) \tag{1.8}$$

where $P(e_j \mid e'_j)$ is the user's assessment of the probability that the evidence e_j is true, given the relevant observation e'_j. An *effective likelihood ratio*, $\lambda'_{j,i}$, is calculated from the posterior odds:

$$\lambda'_{j,i} = \frac{O(h_i \mid e'_j)}{O(h_i)} \tag{1.9}$$

The posterior odds for *all* the evidence is then computed as:

$$O(h_i \mid e'_1, e'_2, \ldots e'_m) = O(h_i) \prod_{j=1}^{m} \lambda'_{j,i} \tag{1.10}$$

Equation 1.8, however, requires a modification, because it over-constrains the input requested from the user. In fact, the user must specify:

- $O(h_i)$, the prior odds on h_i from which $P(h_i)$ can be derived
- $\lambda_{j,i}$, the measure of sufficiency from which $P(h_i \mid e_j)$ can be derived
- $\lambda^*_{j,i}$, the measure of necessity from which $P(h_i \mid \neg e_j)$ can be derived
- $O(e_j)$ the prior odds on e_j from which $P(e_j)$ can be derived

These requirements are equivalent to specifying a line in the space $[\ P(e \mid e'), P(h_i \mid e')]$ by specifying *three* points:

$$(0, P(h_i \mid \neg e_j)),\ (P(e_j), P(h_i)),\ (1, P(h_i \mid e_j))$$

The modification adopted in this approach to prevent the user's inconsistencies is to change equation 1.8 into a piece-wise linear function defined by two line segments passing through the above three points [DHN76].

In an analysis of this approach, Pednault, Zucker, and Muresan [PZM81] concluded that for the cases of *more than two* hypotheses, assumptions 1.4 and 1.5, requiring conditional independence of the evidence both under the hypotheses and their negation, were inconsistent with assumptions 1.2 and 1.3, requiring an exhaustive and mutually exclusive space of hypotheses. Specifically, Pednault proved that, under these assumptions, no probabilistic update could take place, i.e.:

$$P(e_j \mid h_i) = P(e_j \mid \neg h_i) = P(e_j)\ \ \forall i, j \tag{1.11}$$

However, Glymour [Gly85] obtained a pathological counter-example to Pednault's statement (equation 1.11), finding a fault in the original proof of Hussain's theorem that constituted the basis for Pednault's results. Johnson [Joh86] extended this analysis by first showing that there are also non-pathological counter-examples that refute Pednault's results. However, Jonhson proved that under the same assumptions used in Pednault's work, for every hypothesis h_i there is *at most* one piece of evidence e_j that produces updating for h_i. Further studies done by Cheng and Kashyap [CK86] have also indicated that there are at least $max[0, (m - \lfloor \frac{n}{2} \rfloor)]$ pieces of evidence that are *irrelevant*[2] to *all* the

[2]An evidence e_j is said to be irrelevant to the hypothesis h_i if $P(h_i \mid e_j) = P(h_i)$.

hypotheses in the system. This lower bound is for a system satisfying assumptions (4) and (5), in which n is the number of mutually exclusive exhaustive hypotheses ($n > 2$), and m is the number of evidence. Their conclusion is that assumption 1.5 should be dropped.

Pearl has argued in reference [Pea85] that assumption 1.5, requiring the conditional independence of the evidence under the negation of the hypotheses, is over-restrictive. By discarding this assumption, Pearl has derived new, more promising results. However, the assumption 1.4, requiring the conditional independence of the evidence under the hypotheses, is still required for computational efficiency.

The Bayesian approach has various shortcomings. The assumptions on which it is based are not easily satisfiable, e.g. if the network contains multiple paths linking a given evidence to the same hypothesis, the independence assumptions 1.4 and 1.5 are violated. Similarly, assumptions 1.2 and 1.3, requiring the mutually exclusiveness and exhaustiveness of the hypotheses, are not very realistic: assumption 1.2 would not hold if more than one hypothesis could occur simultaneously and is as restrictive as the single-fault assumption of the simplest diagnosing systems; assumption 1.3 implies that every possible hypothesis is *a priori* known, and it would be violated if the problem domain were not suitable to a close-world assumption. Perhaps the most restrictive limitation of the Bayesian approach is its inability to represent *ignorance* (i.e., non-commitment) as illustrated by its *two-way betting* interpretation [Gil82]. The two-way betting interpretation of the Bayesian approach consists of regarding the assignment of probability p to event A as the willingness of a rational agent to accept any of the two following bets:

- *If you pay me \$ p then I agree to pay you \$ 1 if A is true (for $p \in [0,1]$)*
- *If you pay me \$ (1-p) then I agree to pay you \$ 1 if A is false*

The first bet represents the belief that the probability of A is not larger than p, the second bet represents the belief that the probability of A is not smaller than p.

Instead of being explicitly represented, ignorance is *hidden* in prior probabilities. Further shortcomings are represented by the fact that it is impossible to assign any probability to disjunctions, i.e., to non-singletons, which implies the requirement for a uniform granularity of evidence. This problem is usually solved with an approximation, using the Maximum Entropy Principle (MEP). According to MEP, the probability assigned to the disjunct (a subset of singletons in the sample space) is equally divided among the singletons in the subset. This approximation, however, creates an interpretation of the original information, which may not always been appropriate. Finally, as was pointed out by Quinlan [Qui83], in this approach conflictive information is not detected but simply propagated through the network. Some recent work in Bayesian Belief Networks [CJJN90] has actually provided some distinction between rare cases and conflicting data.

1.3.3 Confirmation Theory (Certainty Factors)

The Certainty Factor (CF) approach [SB75], used in MYCIN, is based on Confirmation Theory. The certainty factor CF(h,e) of a given hypothesis h is the difference between a measure of belief MB(h,e) representing the degree of support of a (favorable) evidence e, and a measure of disbelief MD(h,e) representing the degree of refutation of an (unfavorable) evidence e. MB and MD are monotonically increasing functions that are respectively updated when the new evidence supports or refutes the hypothesis under consideration. The certainty factor CF(h,e) is defined as:

$$CF(h,e) = \begin{cases} 1 & \text{if } P(h) = 1 \\ MB(h,e) & \text{if } P(h \mid e) > P(h) \\ 0 & \text{if } P(h \mid e) = P(h) \\ -MD(h,e) & \text{if } P(h \mid e) < P(h) \\ -1 & \text{if } P(h) = 0 \end{cases} \tag{1.12}$$

The measures of belief MB and measure of disbelief MD could be interpreted as a *relative distance* on a bounded interval. Given an interval $[A,B]$ and a reference point R within the interval, the relative distance $d(X,R)$ between any arbitrary point X within the interval and the reference R can be defined as:

$$d(X,R) = \begin{cases} \frac{(X-R)}{(B-R)} & \text{if } X > R \\ 0 & \text{if } X = R \\ \frac{(R-X)}{(R-A)} & \text{if } X < R \end{cases} \tag{1.13}$$

By making the following substitutions in equation 1.13

$$A = 0 \quad B = 1 \quad R = P(h) \quad X = P(h \mid e)$$

the definition of the measure of belief (MB) and measure of disbelief (MD) can be obtained:

$$MB(h,e) = \begin{cases} \frac{P(h|e)-P(h)}{1-P(h)} & \text{if } P(h \mid e) > P(h) \\ 0 & \text{otherwise} \end{cases} \tag{1.14}$$

$$MD(h,e) = \begin{cases} \frac{P(h)-P(h|e)}{P(h)} & \text{if } P(h \mid e) < P(h) \\ 0 & \text{otherwise} \end{cases} \tag{1.15}$$

The CF was originally interpreted as the relative increase or decrease of probabilities. In fact, from equations 1.12, 1.14, and 1.15, it can be shown that:

$$P(h \mid e) = P(h) + CF(h,e) \cdot [1 - P(h)] \quad \text{for } CF(h,e) \geq 0 \tag{1.16}$$

$$P(h \mid e) = P(h) - \mid CF(h,e) \mid \cdot P(h) \quad \text{for } CF(h,e) \leq 0 \tag{1.17}$$

Too often the CF paradigm has been incorrectly used in reasoning systems, interpreting the CFs as *absolute* rather than *incremental* probability values. The original

interpretation of the CF as a probability ratio, however, can no longer be preserved after the CFs have been aggregated using the heuristic combining functions provided in MYCIN [SB75].

Ishizuka, Fu, and Yao [IFY82], [Ish82] have shown that these combining functions were an approximation of the classical Bayesian updating procedure, in which a term had been neglected. In their analysis it was concluded that the assumption of mutual independence of evidence was required for the correct use of this approach. The original definition of certainty factor is asymmetric and prevents commutativity. Another source of concern in the use of CFs is caused by the normalization of MBs and MDs before their arithmetic difference is computed. This normalization *hides* the difference between the cardinality of the set of supporting evidence and that of the set of refuting evidence.

Buchanan & Shortliffe [BS84] have proposed a change to the definition of CF and its rules of combination:

$$CF(h,e) = \frac{MB(h,e) - MD(h,e)}{1 - min(MB(h,e), MD(h,e))} \tag{1.18}$$

$$CF_{COMBINE}(x,y) = \begin{cases} x + y - xy & \text{for } x > 0,\ y > 0 \\ \frac{x+y}{1-min(|x|,|y|)} & \text{for } x < 0,\ y > 0 \text{ or } x > 0,\ y < 0 \\ -CF_{COMBINE}(-x,-y) & \text{for } x < 0,\ y < 0 \end{cases} \tag{1.19}$$

where

$$CF(h,e_1) = x \text{ and } CF(h,e_2) = y$$

This new definition avoids the problem of allowing a single piece of negative (positive) evidence to overwhelm several pieces of positive (negative) evidence. However, it has even less theoretical justification or interpretation than the original formulae.

Recently, Heckerman [Hec86] has derived a new definition for the CF that does allow commutativity and has a consistent probabilistic interpretation. The new definition is:

$$CF(h,e) = \frac{P(h \mid e) - P(h)}{P(h \mid e)[1 - P(h)] + P(h)[1 - P(h \mid e)]} \tag{1.20}$$

There are still numerous serious problems that characterize this approach: the semantics of the CF, i.e., the interpretation of the number (ratio of probability, combination of utility values and probability); the assumptions of independence of the evidence; and the inability of distinguishing between ignorance and conflict, both of which are represented by the assignment (CF = 0).

This type of representation of uncertainty has also been advocated by Rich [Ric83] as an alternative to default reasoning. In her work, Rich claims that default reasoning could actually better be interpreted as likelihood reasoning, providing a uniform representation for statistical, prototypical and definitional facts.

1.3.4 Bayesian Belief Networks

An efficient propagation of belief on Bayesian Networks has been originally proposed by J. Pearl [Pea82]. In his work, Pearl describes an efficient updating scheme for trees and, to a lesser extent, for poly-trees [Pea88a]. However, as the complexity of the graph increases from trees to poly-trees to general graphs, so does the computational complexity.

The complexity for trees is $O(n^2)$ where n is the number of *values per node* in the tree.

The complexity for poly-trees is $O(K^m)$ where K is the number of *values per parent node* and m is the number of *parents per child.* This number is the size of the table attached to each node in the poly-tree. Since the table must be constructed *manually* (and updated automatically), it is reasonable to expect it to be small.

However, the complexity of mulit-connected graphs is $O(K^n)$ where K is the number of *values per node* and n is the size of the largest non-decomposable subgraph.

To handle such complexity, techniques such as *moralization and propagation in a tree of cliques* [LD88] and *loop cutset conditioning* are typically used to decompose the original problem (graph) into a set of smaller problems (subgraphs). When this problem decomposition process is not possible, exact methods must be abandoned in favor of approximate methods. Among these methods the most common are *clustering, bounding conditioning, and simulation techniques (logic samplings and Markov simulations).* See figure 1.1.

1.3.5 Dempster-Shafer (Belief Theory)

The Belief Theory, proposed by Shafer [Sha76], was developed within the framework of Dempster's work on upper and lower probabilities induced by a multivalued mapping[3]

In this context, the lower probabilities have been identified as epistemic probabilities and associated with a degree of belief. This formalism defines certainty as a function that maps subsets of a space of propositions on the [0,1] scale. The sets of partial beliefs are represented by mass distributions of a unit of belief across the propositions in . This distribution is called *basic probability assignment (bpa).* The total certainty over the space is 1. A non-zero *bpa* can be given to the entire space to represent the degree of ignorance. Given a space of propositions , referred to as *frame of discernment*, a function $m : 2 \rightarrow [0,1]$ is called a *basic probability assignment* if it satisfies the following three conditions:

[3]The *one-to-many* nature of the mapping is the fundamental reason for the inability of applying the well-known theorem of probability that determines the probability density of the image of *one-to-one* mappings. In fact, given a differentiable strictly-increasing or strictly-decreasing *function* ϕ on an interval I, and a continuous random variable X with a density f, such that $f(x) = 0$ for any x outside I, then the density function g can be computed as:

$$g(y) = f(x)\left|\frac{dx}{dy}\right|, \; y \in \phi(I) \; and \; x = \phi^{-1}(y)$$

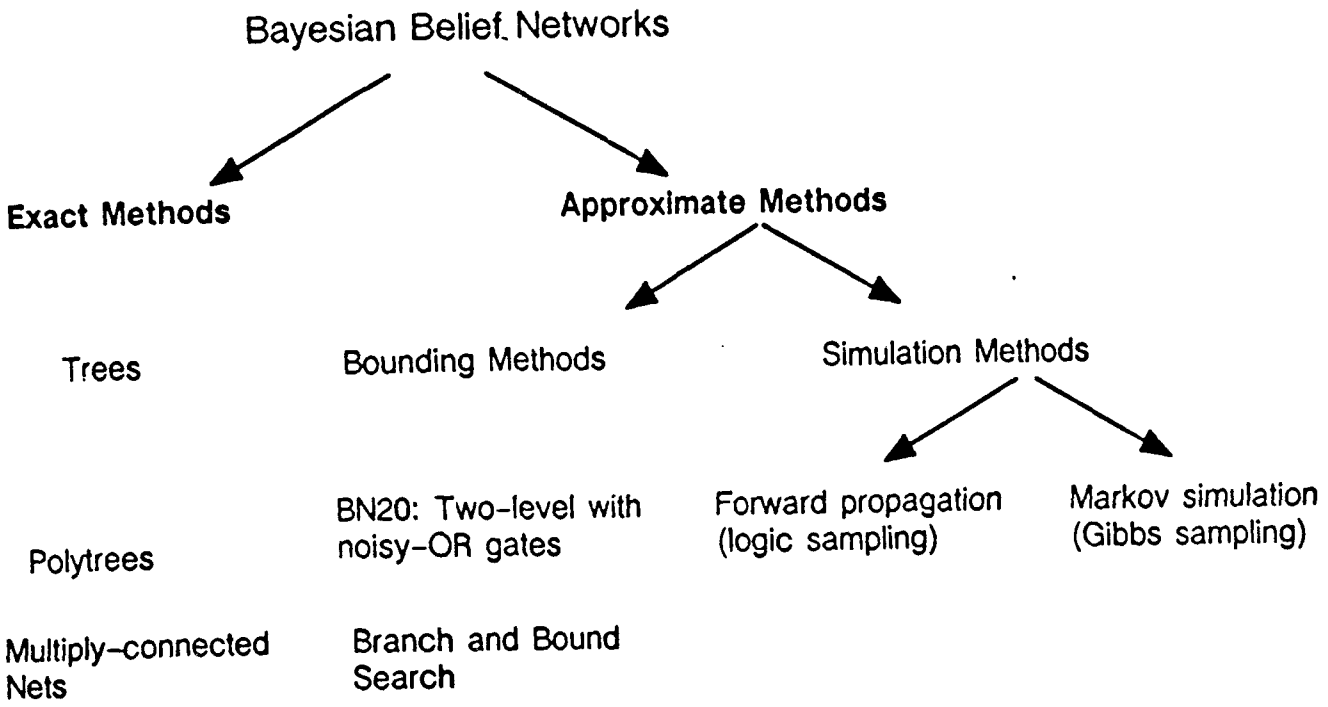


Figure 1.1: Taxonomy of Inference Mechanisms for Bayesian Belief Networks

$$m(\phi) = 0 \quad \text{where } \phi \text{ is the empty set} \tag{1.21}$$

$$0 < m(A) < 1 \tag{1.22}$$

$$\sum_{A \subseteq} m(A) = 1 \tag{1.23}$$

The certainty of any proposition B is then represented by the interval $[Bel(B), P^*(B)]$, where $Bel(B)$ and $P^*(B)$ are defined as:

$$Bel(B) = \sum_{x \subseteq B} m(x) \tag{1.24}$$

$$P^*(B) = \sum_{x \cap B \neq \phi} m(x) \tag{1.25}$$

From the above definitions the following relation can be derived:

$$Bel(B) = 1 - P^*(\neg B) \tag{1.26}$$

If m_1 and m_2 are two *bpa*s induced from two *independent* sources, a third *bpa*, $m(C)$, expressing the pooling of the evidence from the two sources, can be computed by using Dempster's rule of combination:

$$m(C) = \frac{\sum_{A_i \cap B_j = C} m_1(A_i) \cdot m_2(B_j)}{1 - \sum_{A_i \cap B_j = \phi} m_1(A_i) \cdot m_2(B_j)} \tag{1.27}$$

Dempster's rule of combination normalizes the intersection of the bodies of evidence from the two sources by the amount of *non-conflictive evidence* between the sources. This amount is represented by the denominator of the formula.

There are two problems with the Belief Theory approach. The first problem stems from *computational* complexity: in the general case, the evaluation of the degree of belief and upper probability requires time exponential in ||, the cardinality of the hypothesis set (frame of discernment). This is caused by the need of (possibly) enumerating all the subset and superset of a given set. Barnett [Bar81] showed that, when the frame of discernment is discrete (and simple support functions are used), the computational-time complexity could be reduced from exponential to linear by combining the belief functions in a simplifying order. Strat [Str84] proved that the complexity could be reduced to $O(n^2)$, where n is the number of atomic propositions, i.e., intervals of unit length, when the frame of discernment is continuous. In both cases, however, these results were achieved by introducing various assumptions about the type and structure of the evidence to be combined and about the hypotheses to be supported. As a result, in addition to the requirements of mutual exclusive hypotheses and independent evidence which are needed by this approach, the following constraints must be included: for the case of discrete frame of discernment, each piece of evidence is assumed to support only a *singleton* proposition or its negation rather than disjunctions of propositions (i.e., propositions with larger granularity); for the case of continuous frame of discernment, only *contiguous* intervals along the number line can be included in the frame of discernment and thus receive support from the evidence.

The second problem in this approach results from the *normalization* process present in both Dempster's work and Shafer's. Zadeh [Zad84b] [Zad85a] has argued that this normalization process can lead to incorrect and counter-intuitive results. By removing the conflictive parts of the evidence and normalizing the remaining parts, important information is discarded rather than being dealt with adequately. A proposed solution to this problem is to avoid completely the normalization process by maintaining an explicit measure of the amount of conflict and by allowing the remaining information to be *subnormal* (i.e., $Bel() < 1$). Zadeh [Zad85a] has proposed a test to determine the conditions of applicability of Dempster's rule of combination. Dubois and Prade [DP85] have also shown that the normalization process in the rule of evidence combination creates a sensitivity problem, where assigning a zero value or a very small value to a *bpa* causes very different results. It should be noted that this behavior also occurs in other probabilistic schemes, where the assignment of a value of zero to a prior probability would prevent any subsequent updating.

Ginsberg [Gin84] has proposed the use of the Dempster-Shafer approach as an alternative to non-monotonic logic. This suggestion is an extension to Rich's idea of interpreting default reasoning as likelihood reasoning [Ric83]. In his work, Ginsberg provides a rule for propagating the lower and upper bounds through a reasoning chain or graph. His result is based on the interpretation of a production rule as a conditional probability rather than as a material implication. Smets [Sme81], [Sme88] has further explained the relations between belief functions, plausibilities, necessities, and possibilities and has extended Dempster's concepts to handle the case when the evidence is a fuzzy set [Zad65].

Evidential Reasoning

Evidential Reasoning, proposed by Garvey, Lowrance, and Fischler [GLF81], [LG83], [LGS86] adopts the evidential interpretation of the degrees of belief and upper probabilities. Fundamentally based on Dempster-Shafer's theory (as described in Subsection 1.3.5), this approach defines the likelihood of a proposition A as a subinterval of the unit interval [0,1]. The lower bound of this interval is the degree of *support* of the proposition, S(A), and the upper bound is its degree of *plausibility*, Pl(A). The likelihood of a proposition A is written as $A_{[S(A),Pl(A)]}$. The following sample of interval-valued likelihoods illustrates the interpretation provided by this approach:

$A_{[0,1]}$	No knowledge at all about A
$A_{[0,0]}$	A is false
$A_{[1,1]}$	A is true
$A_{[.3,1]}$	The evidence partially supports A
$A_{[0,.7]}$	The evidence partially supports ¬ A
$A_{[.3,.7]}$	The evidence simultaneously provides partial support for A and ¬ A
$A_{[.3,.3]}$	The probability of A is exactly 0.3

Given two statements $A_{[S(A),Pl(A)]}$ and $B_{[S(B),Pl(B)]}$, the set of inference rules corresponding to the logical operations on these statements are defined [GLF81] as:

$$\text{INTERSECTION: } AND(A,B)_{[max(0,S(A)+S(B)-1),min(Pl(A),Pl(B))]} \tag{1.28}$$

$$\text{UNION: } OR(A,B)_{[max(S(A),S(B)),min(1,Pl(A)+Pl(B))]} \tag{1.29}$$

$$\text{NEGATION: } NOT(A)_{[1-Pl(A),1-S(A)]} \tag{1.30}$$

This approach, embodied in GISTER [LGS86], implements Dempster-Shafer (D-S) theory. When distinct bodies of evidence must be pooled, this approach uses the same Dempster-Shafer's techniques, requiring the same normalization process that was criticized by Zadeh.

1.4 Possibilistic Reasoning: Theories and Approaches

1.4.1 Triangular Norm Based Reasoning Systems

Among the possibilistic reasoning techniques, we will discuss the ones based on many-valued logic operators (Triangular norms or T-norms) and the *generalized modus ponens*. These possibilistic techniques have been implemented in RUM [5] and [BW89], a reasoning shell further described in section 1.5.1. For the reader's convenience, RUM's theory is briefly summarized in this section.

Uncertainty in RUM is represented in both facts and rules. Facts are qualified by a degree of confirmation and a degree of refutation. For a fact A, the lower bound of the confirmation and the lower bound of the refutation are denoted by L(A) and L($\neg A$) respectively. As in the case of Dempster's [Dem67] lower and upper probability bounds, the following identity holds: L($\neg A$) = 1 - U(A), where U(A) denotes the upper bound of the uncertainty in A and is interpreted as the amount of failure to refute A. Note that L(A) + L($\neg A$), need not necessarily be equal to 1, as there may be some ignorance about A which is given by (1 - L(A) - L($\neg A$)). The degree of confirmation and refutation for the proposition A can be written as the interval [L(A), U(A)].

RUM provides a natural representation for plausible rules. Rules are discounted by *sufficiency* (s), indicating the strength with which the antecedent implies the consequent and *necessity* (n), indicating the degree to which a failed antecedent implies a negated consequent. Note that conventional strict implication rules are special cases of plausible rules with $s = 1$ and $n = 0$. RUM's inference layer is built on a set of five Triangular norms (T-norms) based calculi [3]. T-norms and T-conorms are two-place functions from [0,1]x[0,1] to [0,1] that are monotonic, commutative and associative. They are the most general families of binary functions which satisfy the requirements of the conjunction and disjunction operators respectively. Their corresponding boundary conditions satisfy the truth tables of the logical AND and OR operators. Five uncertainty calculi based on the following five T- norms are used in RUM:

$$T_1(a,b) = max(0, a+b-1)$$

$$T_{1.5}(a,b) = (a^{0.5} + b^{0.5} - 1)^2 \quad \text{if } (a^{0.5} + b^{0.5}) \geq 1$$
$$= 0 \quad \text{otherwise}$$

$$T_2(a,b) = ab$$

$$T_{2.5}(a,b) = (a^{-1} + b^{-1} - 1)^{-1}$$

$$T_3(a,b) = min(a,b)$$

Their corresponding DeMorgan dual T-conorms, denoted by $S_i(a,b)$, are defined as

$$S_i(a,b) = 1 - T_i(1-a, 1-b)$$

These five calculi provide the user with an ability to choose the desired uncertainty calculus starting from the most conservative (T_1) to the most liberal (T_3). T_1 (T_3) is the most conservative (liberal) T-norm in the sense that for the same input certainty ranges of facts and rule sufficiency and necessity measures, T_1 (T_3) shall yield the minimum (maximum) degree of confirmation of the conclusion. For each calculus (represented by the above five T-norms), the following four operations have been defined in RUM:

Antecedent Evaluation. To determine the aggregated certainty range [b, B] of the n clauses in the antecedent of a rule, when the certainty range of the ith clause is given by $[b_i, B_i]$:

$$[b, B] = [T_i(b_1, b_2, \ldots, b_n), T_i(B_1, B_2, \ldots, B_n)]$$

Conclusion Detachment: Modus Ponens. To determine the certainty range, [c,C] of the conclusion of a rule, given the aggregated certainty range, [b,B] of the rule premise and the rule sufficiency, s and rule necessity, n:

$$[c, C] = [T_i(s, b), 1 - (T_i(n, (1 - B)))]$$

Conclusion Aggregation. To determine the consolidated certainty range [d, D], of a conclusion when it is supported by m ($m > 1$) paths in the rule deduction graph, i.e., by m rule instances, each with the same conclusion aggregation T-norm operator. If $[c_i, C_i]$ represents the certainty range of the same conclusion inferred by the ith proof path (rule instance), then

$$[d, D] = [S_i(c_1, c_2, \ldots c_m), S_i(C_1, C_2, \ldots, C_m)]$$

Source Consensus. To determine the certainty range, $[L_{tot}(A), U_{tot}(A)]$ of the same evidence, A, obtained by fusing the certainty ranges, $[L_i(A), U_i(A)]$, of the *ith* information source out of a total of n different possible information sources:

$$[L_{tot}(A), U_{tot}(A)] = [Max_{i=1,\ldots,n} L_i(A), Min_{i=1,\ldots,n} U_i(A)]$$

The theory of RUM is anchored on the semantics of many-valued logics [3]. Unlike other probabilistic systems, RUM's reasoning mechanism is possibilistic. Reference [3] describes a comparison of RUM with other reasoning with uncertainty systems, such as Modified Bayesian [DHN76], Certainty Factors [SB75], [Hec86], Dempster-Shafer [Dem67], [Sha76], and Fuzzy logic [Zad65].

1.5 Technology for Possibilistic Reasoning

We have embedded the theory of possibilitic reasoning in an integrated reasoning system composed of RUM [5], a rich, user-friendly development environment, and RUMrunner, a small and quick run-time system, and translation software to span the two (see Figure 1.2).

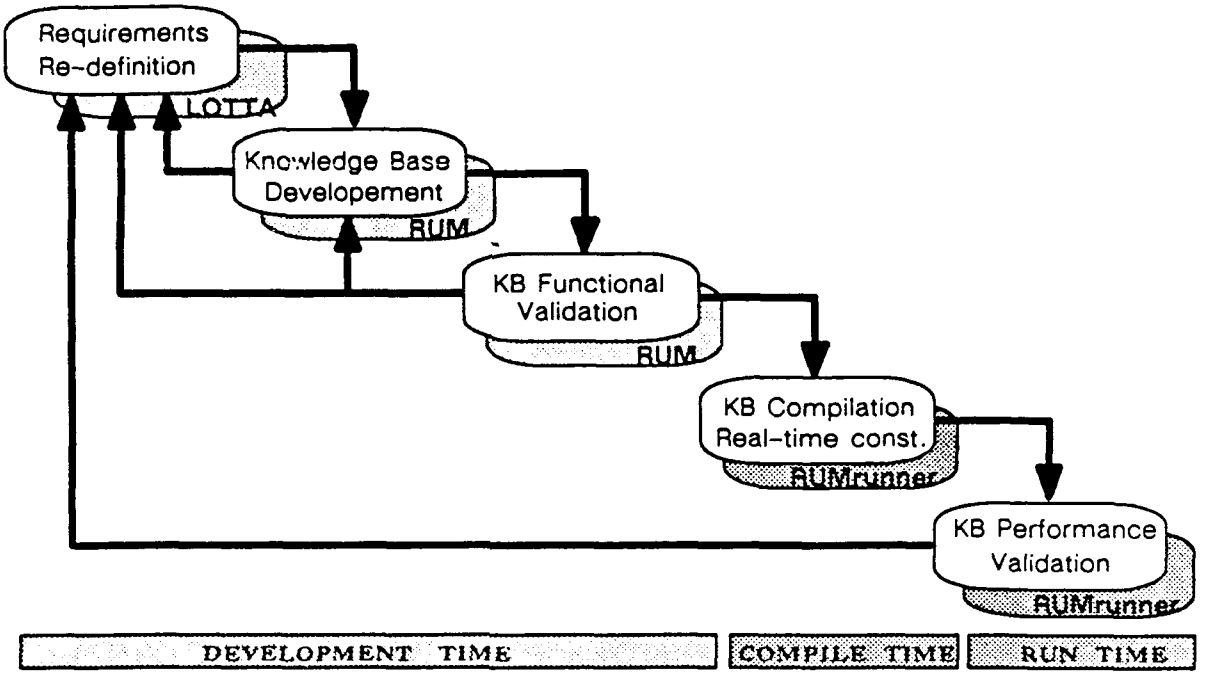


Figure 1.2: Software Engineering with RUM and RUMrunner

1.5.1 Possibilistic Reasoning System: RUM

RUM embodies the theory of plausible reasoning described in the previous section. RUM provides a representation of uncertain information, uncertainty calculi for inferencing, and selection of calculi for inference control. Uncertainty is represented in both facts and rules. A fact represents the assignment of a value to a variable. A rule represents the deduction of a new fact (conclusion) from a set of given facts (premises). Facts are qualified by a degree of *confirmation* and a degree of *refutation*. As we have noted in Subsection 1.4.1, rules are discounted by *sufficiency*, indicating the strength with which the premise implies the conclusion, and *necessity*, indicating the degree to which a failed premise implies a negated conclusion. The uncertainty present in this deductive process leads to considering several possible values for the same variable. Each value assignment is qualified by different uncertainties, which are combined with T-norm based calculi as described in [3] and [4].

RUM's rule-based system integrates both procedural and declarative knowledge in its representation. This integration is essential for solving situation assessment problems, which involve both heuristic and procedural knowledge.

The expressiveness of RUM is further enhanced by two other functionalities: the *context mechanism* and *belief revision*. The context represents the set of preconditions determining the rule's applicability to a given situation. This mechanism provides an efficient screening of the knowledge base by focusing the inference process on small

rule subsets. For instance, in SA, selected rules describe the behavior of friendly planes, while others should only be applied to unfriendly or unidentified ones. The rule's context provides this filtering mechanism.

RUM's belief revision is essential to the dynamic aspect of the classification problem. The belief revision mechanism detects changes in the input, keeps track of the dependency of intermediate and final conclusions on these inputs, and maintains the validity of these inferences. For any conclusion made by a rule, the mechanism monitors the changes in the certainty measures that constitute the conclusion's support. Validity flags are used to reflect the state of the certainty. For example, a flag can indicate that the uncertainty measure is valid, unreliable (because of a change in the support), too ignorant to be useful, or inconsistent with respect to the other evidence.

RUM offers both backward and forward processing. A *lazy evaluation*, running in backward mode, recomputes the certainty measures of the minimal set of facts required to answer a given query. This mode is used when the system or the user decide that they are dealing with time-critical tasks. Breadth-first, forward mode processing recomputes the certainty measures attempting to restore the integrity of the rule deduction graph. This mode is used by the system when time is not critical.

These AI capabilities are used to develop a knowledge base, in conjunction with RUM's software engineering facilities, such as flexible editing, error checking, and debugging. Some of these features, however, are no longer necessary once the development cycle is complete. At run-time, applications do not create new knowledge (facts or rules), because their basic structures have been determined at compile-time. The only run-time requirement is the ability to instantiate rules and facts from their predetermined definitions. By eliminating the development features that are unnecessary at run-time, a real-time AI system can improve upon the algorithms and methodologies used in RUM.

1.5.2 Possibilistic Reasoning System: RUMrunner

The objective of RUMrunner [Pfa87] is to provide a software tool that transforms the customized knowledge base generated during the development phase into a fast and efficient real-time application. RUMrunner provides both the functionality to reason about a broad set of problems and the speed required to properly use the results of the reasoning process. Performance improvements are obtained by implementing all RUM's functionalities with leaner data structures, using Flavors (for the Symbolics version) or *defstructs* (for the Sun version). Furthermore, RUMrunner no longer requires the use of the KEE software, thus it can be run on any Symbolics or Sun workstation with much smaller memory configurations and without a KEE software license. RUMrunner has four major qualities: it provides a meaningful subset of AI techniques; it runs fast; it has the functionality of a real-time system; and it does not require the software engineer to reprogram the application in the target environment.

This goal is achieved by a combination of efforts: the translation of RUM's (development system) complex data structure into simpler, more efficient ones (to reduce overhead); the compilation of the rule set into a compiled network (to avoid run-time

search); the load-time estimation of each rule's execution cost (to determine, at run-time, the execution cost of any given deductive path); and the planning mechanism for model selection (to determine the largest relevant rule subset which could be executed within a given time-budget). Figure 1.3 shows the RUMrunner architecture.

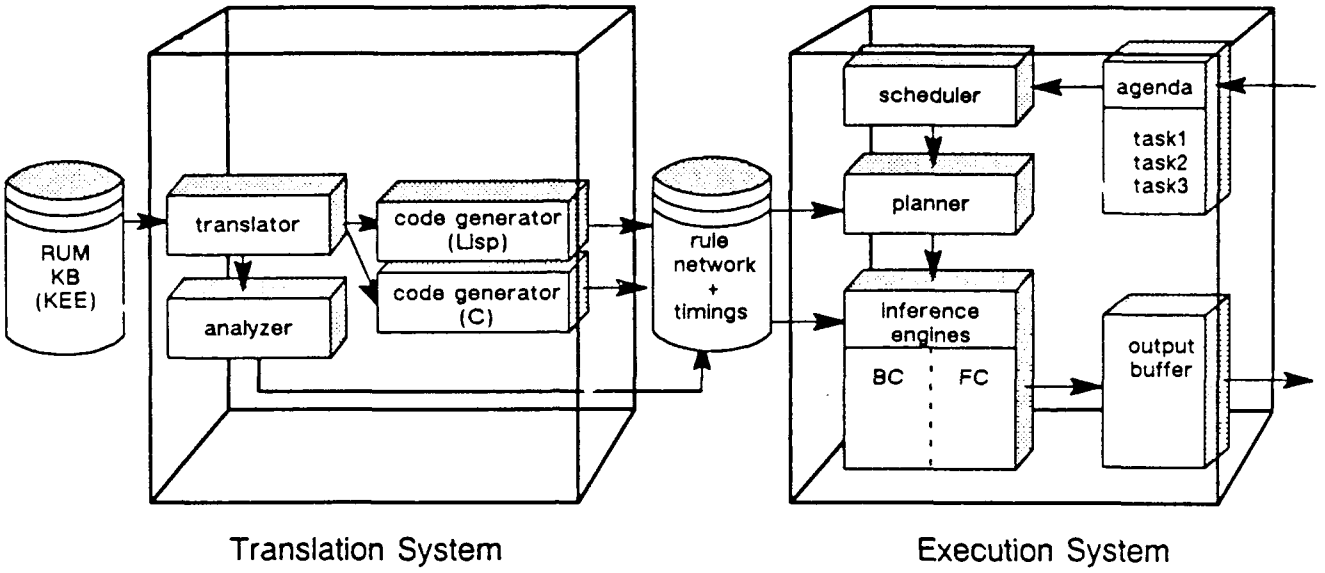


Figure 1.3: RUMrunner Architecture

An agenda mechanism is used to asynchronously receive any number of input tasks (such as backward-chaining on a goal or forward-chaining on a given piece of evidence) from various sources. Each task in the agenda receives a (static) priority number, determining the relative importance of the task with respect to the others. A time deadline, expressed in absolute time, is attached to the task to indicate its urgency (i.e., its expiration time), which is used by the planning mechanisms described below.

A scheduler sorts the tasks by priority and, within the same priority level, by the shortest deadline. The the highest priority task is then scheduled for execution by the forward or backward chainer. [DL87].

The results of these tasks are in turn isolated from external connecting systems via buffers or streams and a layer of interface functions.

External or internal interrupts, with re-entrant reasoning, can supersede the current task. There are three classes of interrupts possible: internal interrupts caused by queries approaching their assigned time deadlines or exceeding other reasoning resources; exter-

nal interrupts caused by queries with higher priority than the one currently addressed; and external interrupts caused by new input data characterized by higher priority than the current query. Since the state of the current knowledge base is dynamically maintained in the knowledge base nodes themselves, any changes to the knowledge base by the interrupting task will be automatically taken into account when the pre-empted task is resumed.

In order to keep track of time requirements and resources available to ensure real-time response, a planning and allocation module provides a control layer on top of the inference mechanisms. Our planning scheme considers a backward chaining query to be a goal which can be satisfied by using various inference paths. Planning for real-time performance involves generating a set of plans (solution paths), evaluating their usefulness and cost, and selecting some or all of them to be used to satisfy the query within a given time deadline. Since the solutions have some associated uncertainty, executing multiple plans may improve the quality (certainty) of the initial result.

The current implementation generates and uses these plans in a fairly simple way, maximizing the total *number* of nodes in the solution set (within the allocated time budget) without considering any further plan attributes. The plan set is fully enumerated and committed to before the task is begun, without any considerations for efficiency or interruptibility.

Our approach for improving this strategy involves imposing an initial ordering on the set of plans, based on several significant measures of each plan's cost and expected benefit. As each plan is executed, in "best-first" order, this initial ordering may be updated as additional values and certainties are inferred, making the planning process more opportunistic. Executing the plans in order of expected overall utility also provides for interruptibility, since a partial answer becomes available as soon as the first plan has been completed.

In summary, RUMrunner takes advantage of the fact that the application has been completely developed and debugged. It provides a minimum of error checking because the application is assumed either to be debugged already, or to be robust enough to handle errors. RUMrunner's time performance in reasoning tasks is partially attributed to the compilation of the knowledge base. As a result of this compilation, new or different rules or units cannot be created in the knowledge base after the translation. Finally, RUMrunner is implemented in Common LISP, thus it can be ported to many machines without requiring any proprietary software. RUMrunner, is further elaborated upon in [Pfa87].

The reasoning technologies described in the previous section have been tested in a variety of applications, such as Pilot's Associate, Submarine Commander Associate, Air Land Battle Management (Division Level Fire Support). These applications are all examples of the dynamic classification paradigm that is described in the following section.

1.5.3 Possibilistic Reasoning System: PRIMO

The most recently developed technology embodying possibilistic reasoning techniques is the Plausible ReasonIng MOdule (PRIMO). Developed as part of the Knowledge-Based System Technology Base in the Strategic Computing Initiative, PRIMO is a reasoning system which integrates the theories of plausible reasoning (based on monotonic rules with degrees of uncertainty) and defeasible reasoning (based on default values supported by nonmonotonic rules). The PRIMO system consists of a representation language which includes declarative specifications of uncertainty and default knowledge, reasoning algorithms, and an application development environment.

In this section we review the theoretical foundations of PRIMO (see [5, BCGS89]) and discuss PRIMO's implementation.

Uncertainty

The uncertainty representation used in PRIMO is based on the semantics of many-valued logics. PRIMO, like its predecessor RUM [5], uses a combination of fuzzy logic and interval logic to represent and reason about uncertainty. This approach has been successfully demonstrated in two DARPA applications: the Situation Assessment Module of Pilot's Associate (Phase I) and the technology demonstration of the Submarine Operational Automation System (Phase I).

PRIMO handles uncertain information by qualifying each possible value assignment to any given propositional variable with an uncertainty interval. The interval's lower bound represents the minimal degree of confirmation for the value assignment. The upper bound represents the degree to which the evidence failed to refute the value assignment. The interval's width represents the amount of ignorance attached to the value assignment. The uncertainty intervals are propagated and aggregated by Triangular-norm-based uncertainty calculi (see [1, 3, SS83, 6]). The uncertainty interval constrains intervals of subsequent, dependent values.

Incompleteness

PRIMO handles incomplete information by evaluating non-monotonic justified (NMJ) rules. These rules are used to express the knowledge engineer's preference in cases of total or partial ignorance regarding the value assignment of a given propositional variable. The NMJ rules are used when there is no plausible evidence (to a given numerical threshold of belief or certainty) to infer that a given value assignment is either true or false. The conclusions of NMJ rules can be retracted by the belief revision system, when enough plausible evidence is available.

PRIMO uses the numerical certainty values generated by plausible reasoning techniques to quantitatively distinguish the admissible extensions generated by defeasible reasoning techniques. The method selects a *maximally consistent extension* (see [BCGS89]) given all currently available information.

For efficiency considerations some restrictions are placed on the language in which one can express PRIMO rules. The monotonic rules are non-cyclic Horn clauses, and are maintained by a linear belief revision algorithm operating on a rule graph. The NMJ rules can have cycles, but cannot have disjunctions in their conclusions.

By identifying sets of NMJ rules as strongly connected components (SCC's), we can decompose the rule graph into a directed acyclic graph (DAG) of nodes, some of which are SCCs with several input edges and output edges. PRIMO contains algorithms to efficiently propagate uncertain and incomplete information through these structures at run time. Treating the SCCs independently can result in a significant performance improvement over processing the entire graph. However, this heuristic may result in loss of correctness in the worst case. These algorithms require finding satisfying assignments for nodes in each SCC, and are thus NP-hard in the unrestricted case. We can achieve tractability by restricting the size and complexity of the SCC's, precomputing their structural information, and using run-time evaluated certainty measures to select the most likely extension.

A more detailed description of PRIMO can be found in Sections 9 and 10.

1.6 Desiderata for Reasoning with Uncertainty

In the previous section we have discussed probabilistic and possibilistic reasoning technologies. In this section we will compare them against a set of requirements. This idea was first proposed by Quinlan, who suggested a list of four requirements to illustrate the shortcomings of the Bayesian and Confirmation theory approaches and to compare them with INFERNO, his proposed approach to uncertain inference [Qui83]. The requirements proposed by Quinlan were:

- "An inference system should not depend on any assumptions about the probability distributions of the propositions".
- "It should be possible to assert common relationships between propositions ... when the relationships are indeed known".
- "It should be possible to posit information about any set of propositions and observe the consequences for the system as a whole"
- "If the information provided to the system is inconsistent, this fact should be made evident along with some notion of alternative ways that the information could be made consistent".

Quinlan's work has been inspirational in the development of the following *desiderata*, which subsumes and extends Quinlan's initial list. The proposed desiderata describes the requirements to be satisfied by the ideal formalism for representing uncertainty and making inference with uncertain information. To be consistent with the organizing principle typical of automated reasoning systems, the desiderata is subdivided into the same three layers of *Representation*, *Inference* and *Control*.

Representation Layer

1. There should be an explicit representation of the *amount* of evidence for *supporting* and for *refuting* any given hypothesis.

2. There should be an explicit representation of the information about the evidence, i.e., *meta-information*, such as the evidence source, the reasons for supporting and for refuting a given hypothesis, etc. This meta-information will be used in the control layer to remove conflicting pieces of evidence provided by different sources.

3. The representation should allow the user to describe the uncertainty of information at the available level of detail, ranging from singletons to any subset of the universe of discourse. We will refer to this property as *heterogeneous information granularity*.

4. There should be an explicit representation of *consistency*. Some measure of consistency or compatibility should be available to detect trends of potential conflicts and to identify essential contributing factors in the conflict.

5. There should be an explicit representation of *ignorance* to allow the user to make *non-committing* statements, i.e., to express the user's lack of conviction about the certainty of *any* of the available choices or events. Some measure of ignorance, similar to the concept of entropy, should be available to guide the gathering of discriminant information.

6. The representation must be, or at least must appear to be natural to the user to enable him/her to *describe uncertain input* and to *interpret uncertain output*. The representation must also be natural to the expert to enable him/her to elicit *consistent* weights representing the strength of the implication of each rule.

Inference Layer

7. The combining rules should not be based on global assumptions of *evidence independence*.

8. The combining rules should not be based on global assumptions of *hypotheses exhaustiveness* and *exclusiveness*.

9. The combining rules should maintain the *closure* of the syntax and semantics of the representation of uncertainty.

10. Any function used to propagate and summarize uncertainty should have clear semantics. This is needed both to maintain the semantic closure of the representation and to allow the control layer to *select* the most appropriate combining rules.

Control Layer

11. There should be a clear distinction between a *conflict* in the information (i.e., violation of consistency), and *ignorance* about the information. To solve the conflict, the controller (meta-reasoner) must retract one or more elements of the conflicting set of evidence. To remove the ignorance, the controller must select a (retractable) default value or tag the information with an assumption.

12. The *traceability* of the aggregation and propagation of uncertainty through the reasoning process must be available to resolve conflicts or contradictions, to explain the support of conclusions, and to perform meta-reasoning for control.

13. It should be possible to make pairwise comparisons of uncertainty since the induced *ordinal or cardinal ranking* is needed for performing any kind of decision-making activities.

14. It should be possible to *select* the most appropriate combination rule by using a declarative form of control (i.e., by using a set of context dependent rules that specify the selection policies).

1.6.1 Evaluation of the Approaches

The above desiderata was used to guide the development of RUM and PRIMO.

APPROACH	*REPRESENTATION*						*INFERENCE*				*CONTROL*			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Modified Bayesian	N	N	N	N	N	Y	N	N	Y	Y	N	N	Y	N
Confirmation	N	N	N	N	Y	N	N	Y	N	N	N	N	N	N
Dempster-Shafer	Y	N	Y	Y	Y	Y	N	N	Y	Y	Y	N	Y	N
Probability Bounds	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
Fuzzy necessity/possibility	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
Evidence Space	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
RUM/PRIMO	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Reasoned Assumptions	N	Y	N	Y	N	Y	Y	Y	Y	Y	N	Y	N	Y
Endorsements	N	Y	N	Y	N	Y	Y	Y	Y	Y	N	Y	N	Y

Figure 1.4: Evaluation of Uncertainty Approaches Against the Desiderata

Table 1.4 summarizes the evaluation of the formalisms discussed in the previous section against this desiderata. The order in which the formalisms appear in the table reflects their numeric or non-numeric nature: the numeric formalisms are listed above RUM/PRIMO, the non-numeric ones are shown below it. RUM/PRIMO is considered a hybrid as it uses both numeric and symbolic information.

Evaluation of PRIMO

This part of the section illustrates how PRIMO meets the majority of the requirements described in the above desiderata.

Representation Layer

1. *Explicit representation of the* amount *of evidence for* supporting *and for* refuting *any given hypothesis.*

   Yes: Any evidence A has an associated unit with the numerical interval $[L(A), U(A)]$ that capture the amounts of support and refutation. The boundaries of this interval can take numerical or linguistic probability values.

2. *Explicit representation of the information about the evidence, i.e.,* meta-information

   Yes: PRIMO's representation layer contains symbolic information. For input nodes PRIMO stores the source of the evidence and its credibility. for intermediate nodes, PRIMO maintains the logical support and the amount of discounting used on the path leading to the node. This information is used by the control layer to efficiently implement the nodes belief revision and to resolve ignorance or conflicts among various sources.

3. *Heterogeneous information granularity*

   No: PRIMO rule representation language only allows the user to have singletons in the right-hand side of each rule. Therefore, PRIMO can only qualify the belief of value assignments to single variables, rather than to arbitrary subsets of variables.

4. *Explicit representation of consistency*

   Yes: A violation of the constraint $L(A) \leq U(A)$ will detect the occurrence of an inconsistency. In this case, a simple measure of the inconsistency is given by the difference $L(A) - U(A)$. This measure of consistency is needed to detect trends of potential conflicts and to identify essential contributing factors in the conflict.

5. *Explicit representation of ignorance*

   Yes: The difference between the upper and lower bound, i.e. $U(A) - L(A)$ is a measure of the amount of lack of commitment or ignorance. Thus the width of the interval is used to express the user or system's lack of conviction about the certainty of *any* of the available choices or events.

6. *Natural interpretation of the representation to the user and the expert*

   Yes: Linguistic probabilities used by the user/expert to assess likelihood estimates provide a natural, easy to calibrate uncertainty representation. In the internal parametric representation the linguistic probabilities are mapped into fuzzy intervals. The parametric representation provides a common and efficient formalism in which more

precise estimates, such as crisp probabilities or crisp intervals, can be used in conjunction with the linguistic probabilities.

Inference Layer

7. *Removing global assumptions of evidence independence from combining rules*

   Yes: The calculus selection is driven by *local* contextual information. In those contexts where the evidence is independent, the appropriate T-norm, such as T_2, will be selected.

8. *Removing global assumptions of hypotheses exhaustiveness and exclusiveness from combining rules*

   Yes: No global assumptions are used in the calculus selection. This particular assumption is not needed since no normalization process takes place.

9. *Maintaining syntactic and semantic closure of the representation under the combining rules*

   Yes: The T-norm based calculi maintain the semantic closure of the data. A closed-form solution to the extension principle problem provides a set of formulae that maintain the closure of the parametric representation used to internally characterize the information. The linguistic probabilities used as an option in describing the input from the user/expert are represented in the same parametric form. At the end of the reasoning process the parametric form can be expressed again in term of linguistic probabilities by using the linguistic approximation process.

10. *Clear semantics of the combining rules*

    Yes: Any function used to propagate and summarize uncertainty should have clear semantics. This is needed both to maintain the semantic closure of the representation and to allow the control layer to *select* the most appropriate combining rules. The uncertainty calculi used in the inference layer have distinct properties and meanings. These characteristics are used in the control layer to define a set of context-dependent selection policies.

    The uncertainty calculi are ordered from a lower bound, the calculus based on T_1, to an upper bound, the calculus based on T_3. This ordering can be interpreted as a transition from negative correlation (T_1) to positive correlation (T_3). Another possible interpretation of the meaning of the proposed calculi is to consider their ordering as a transition from a pessimistic (risk-avoidance) attitude (T_1), to an optimistic (gambling) attitude (T_3).

Control Layer

11. *Clear distinction between conflict and ignorance*

Yes: Conflict and ignorance of the uncertainty measure are mutually exclusive: conflict occurs when $L(A) > U(A)$, ignorance is present when $L(A) < U(A)$.

This distinction is important since the controller must react differently to each case: to solve the conflict, the meta-reasoner must retract one or more elements of the conflicting set of evidence; to remove the ignorance, the controller must select a (retractable) default value or tag the information with an assumption.

12. *Traceability of the uncertainty aggregation and propagation*

Yes: The separation between the inference and the control layer provides a mechanism for tracing the selection and application of uncertainty calculi. This book-keeping activity can then be used by a Reason Maintenance System (RMS) to update the uncertainty values that exhibit any dependency from a modified piece of evidence.

A first implementation of the belief revision of the uncertain information has been implemented in the control layer of PRIMO'S Rule System. For any (propositional) conclusion made by a rule instance, the belief revision mechanism monitors the changes in the certainty measures attached to the variable node that constitute the conclusion's support or the changes in the calculus used to compute the conclusion certainty measure. Validity flags are inexpensively propagated through the rule deduction graph.

The *traceability* of the aggregation and propagation of uncertainty through the reasoning process must be available to resolve conflicts or contradictions, to explain the support of conclusions, and to perform meta-reasoning for control.

13. *Pairwise comparisons based on an ordinal or cardinal ranking.*

Yes: Various ordering functions can be used to rank two pieces of evidence on the basis of their uncertainty measures. The simplest (complete) ordering is obtained by selecting the evidence with the *highest lower bound*, i.e., A is preferred to B if $L(A) > L(B)$. A partial ordering function is obtained by selecting A over B if $[L(A), U(A)] > [L(B), U(B)]$. Alternatively, more complex partial ordering functions could also be defined.

14. *Selecting the most appropriate combining rule*

Yes: The calculi selection is explicit and programmable by using a declarative form of control, i.e., a set of context dependent rules that specify the selection policies.

1.7 Dynamic Classification Problems: Situation Assessment and Tactical Planning

We have described, analyzed and classified various approaches to reasoning with uncertainty according to their complexity, semantics, and computational cost. Now we want to focus on the main functionalities of a class of problems known as *classification problems*.

The *Classification Problem* (CP) was first introduced by Clancey [Cla84] in 1984, and consists of recognizing a situation from a collection of data and selecting the best action in accordance with some objectives. The classification problem has a recurrent solution structure:

1. A collection of data, generated from several sources, is interpreted as a predefined pattern.
2. The recognized pattern is mapped into a set of possible solutions.
3. One of these solutions is selected as the most appropriate for the given case.

This process was considered a *static* classification problem, since the input data were assumed to be invariant over time, or at least invariant over the time required to obtain the solution.

A more interesting and challenging case is the *Dynamic Classification Problem* (DCP), originally described in [BW88], in which the environment from which data are collected changes at a rate comparable with the time required to obtain a refined solution, requiring real-time response. The characteristic structure of this class of dynamic classification problems is illustrated in Figure 1.5.

1.7.1 Situation Assessment

As part of the DCP, we will first describe Situation Assessment. Given a platform (submarine) in a potentially hostile environment, the process of Situation Assessment consists of the following tasks:

1. Sensor data is collected and consolidated from various sources, and fused into related *tracks* representing individual *contacts*. This process constitutes what is generally known as *information fusion* or *situation description*.
2. For selected *interesting* contacts, the analysis is extended to determine the contacts' formation, use of special equipment, and maneuvering. This information is used with the knowledge of the opponent's doctrines and rules of engagement to determine if the contact is *aware of ownship*, to analyze the contact's *behavior* and to infer its probable *intent and mission mode*. These intents are then used to derive a *threat assessment*, which is in turn combined with our mission description, the contacts' weapons type, and their perceived weapon-range, to estimate the contacts' *target* value. These activities constitute the *retrospective* component of SA.
3. The current assessment of the situation is projected using a short-term horizon, to estimate the contacts' future position, course, intent, threat, target and to determine potentially dangerous or interesting events, before they occur and determining ownship's *vulnerability*. This constitutes the *prospective* component of SA.

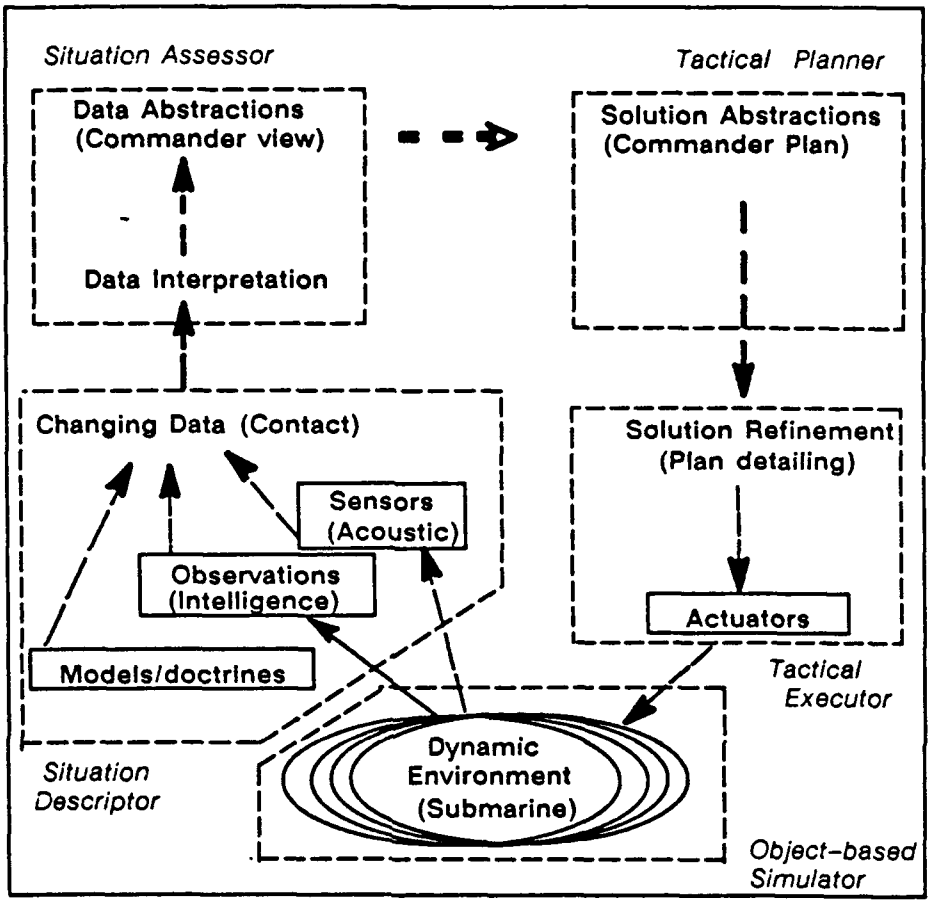


Figure 1.5: The Dynamic Classification Problem

4. Finally, the output of the Situation Assessment module is sent to a Tactical Planner (TP) module, which determines the best course of action to follow. As a result, a *plan monitoring* requests, defined by a set of geometrical or behavioral constraints on contacts or events, may be sent back to the SA module, which will monitor these constraints and notify TP of any existing or potential violations.

1.7.2 The Role of Uncertainty in SA-TP

To analyze the role and the impact of uncertainty management in SA and TP we will refer to the process illustrated in Figure 1.6 (we assume that TP is implemented using case-based planning technology).

This figure describes the need to achieve a trade-off between accuracy/coverage and computational cost. We can observe that *multiple scenes* are generated by the Situation Interpretation module, *multiple interpretations* for each scene are provided by the Sit-

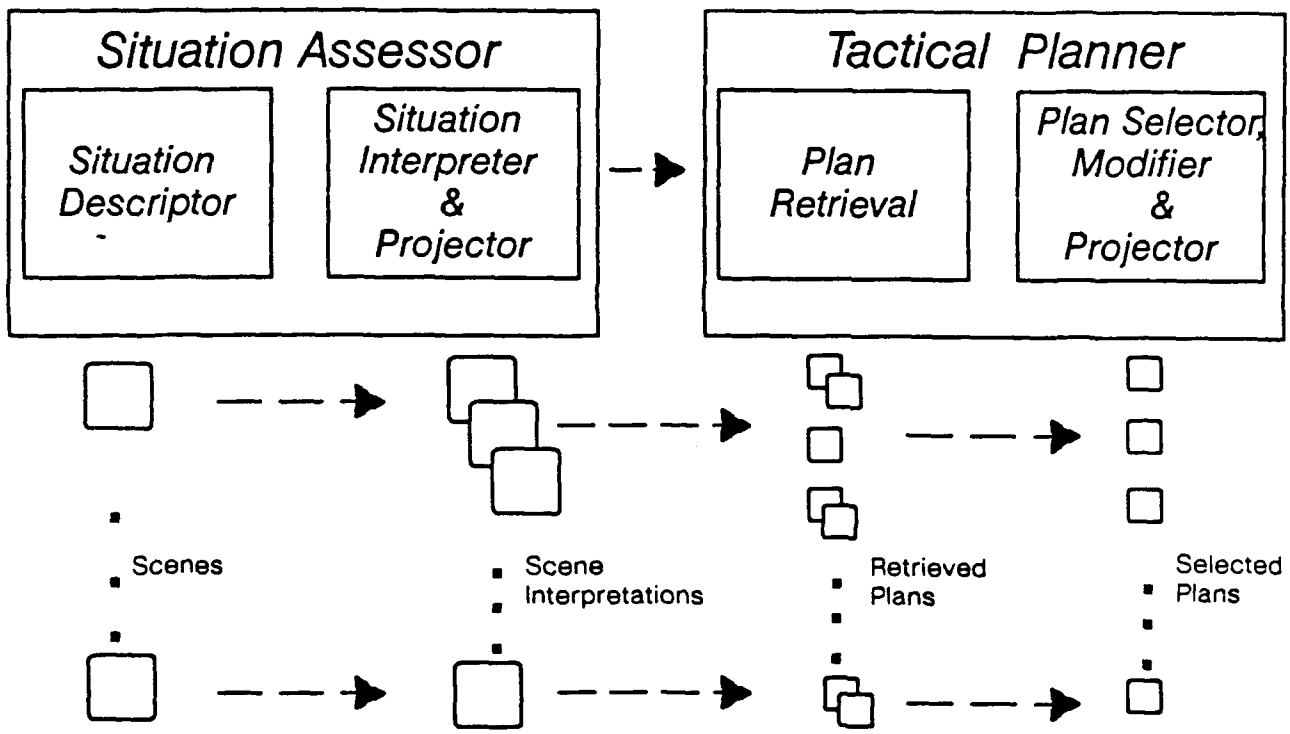


Figure 1.6: Ambiguity and Uncertainty in SA and TP

uation Interpreter module and *multiple plans*, developed for similar interpretations and indexed by a set of abstract features, are retrieved by the Plan Retriever Module for each interpretation. Finally, for each case (scene interpretation) a plan is selected, adapted, projected and repaired (if needed). Among all these plans, one is finally selected for execution.

It is clear that under real-time pressure, it is not possible to exhaustively analyze all cases. It is therefore essential to control the number of scenes and scenes interpretations and to focus TP's efforts on the most likely or important interpretations. Thus we suggest to use a figure of merit for each scene, and, subsequently for each scene interpretation, to control this potential information explosion. The following example illustrates the use of Dempster-Shafer in deriving such a figure of merit for the classification enhancement in a scene.

1.7.3 Example 1: Use of Dempster-Shafer in SA Classification Enhancement

Let us assume that we are collecting information from two independent sensors, referred to as Sensor 1 and Sensor 2. Sensor 1 has detected a contact M-1, determined that it was a submarine, and provided a tentative classification:

```
Submarine of Type A 0.6
```

```
Submarine of Type B 0.3
Some type of submarine 0.1
```

The second sensor has detected a second contact, M-2, determined that it was a torpedo, and provided the following tentative classification:

```
Torpedo of Type X 0.6
Torpedo of Type Y 0.3
Some type of Torpedo 0.1
```

obtain nine possible worlds, as indicated by Figure 1.7.

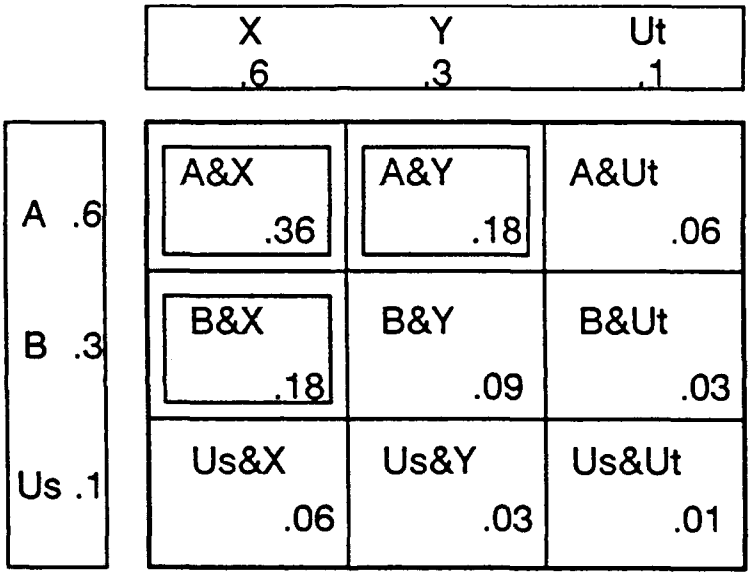


Figure 1.7: Generation of Nine Possible Worlds

The terms Us and Ut in Figure 1.7 refer to the universe of submarines and torpedoes, respectively. From this figure we can observe that only three possible worlds (indicated by the gray boxes) have a combined figure of merit which is greater than 0.1. We can use these figures of merit to rank the possible worlds and to limit its processing as a function of the real-time pressure we may experience. It is interesting to note that we also get a sense for the amount of *coverage* that we are providing with this analysis. The combined figures of merit of the three scenes (A&X),(A&Y),(B&X) give us a coverage of 72% of all possible cases.

Let us now assume that, by looking at some intelligence data base, we discover that submarines of type A typically carry torpedoes of type Y. We can represent this information as: $(A \rightarrow Y)(0.9)$ and $(A \rightarrow Ut)(0.1)$. We can express each implication by its boolean equivalent and have: $(\neg A \cup Y)(0.9)$ and $(\neg A \cup Ut)(0.1)$.

If we now fuse this information with the previous sensor information we have an update which is illustrated in Figure 1.8.

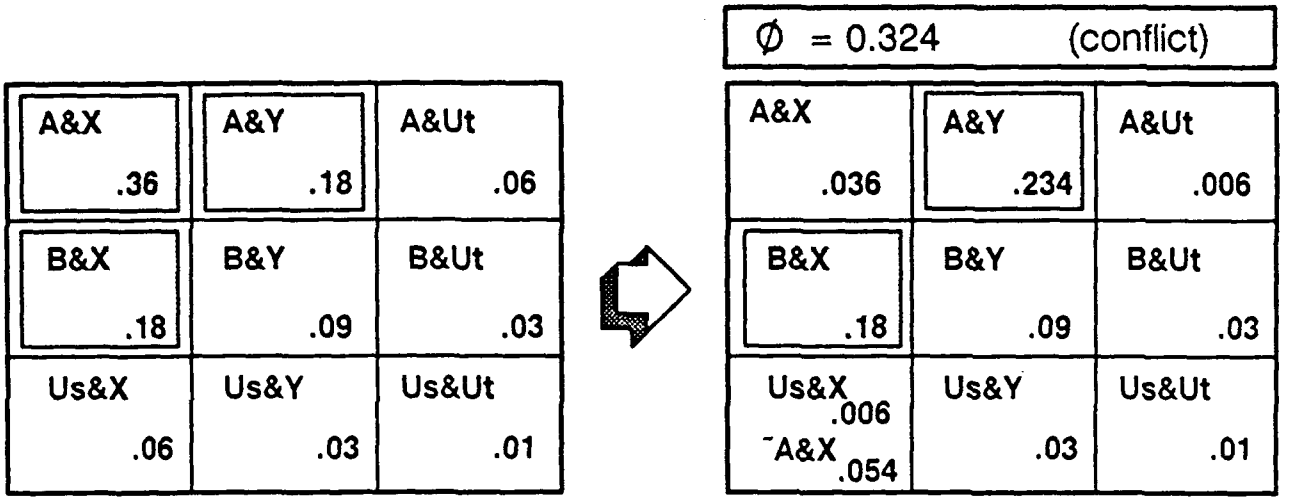


Figure 1.8: Result of the Updating Process

From Figure 1.8, by using equations 1.24 and 1.25 in Section 1.3.5, we can compute the lower and upper bounds of each possible world before and after the update using the intelligence information. This computation is shown in Figure 1.9.

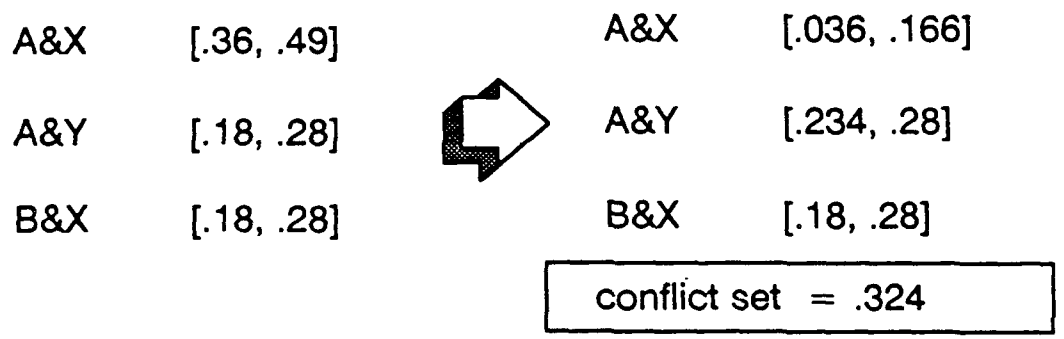


Figure 1.9: Computation of New Bounds

We can observe that the first possible world (A&X) has been dropped below our threshold of 0.1. The second possible world (A&Y) has increased in its amount of belief, while the third one (B&X) has not been affected by the update. We can also observe that there is a substantial amount of conflict affecting the first two possible worlds, which indicates that there is a certain amount of inconsistency between the sensors and the intelligence information. This measure of conflict can be used to normalize the lower and upper bounds, as it is also illustrated in Figure 1.10.

On the other hand, we can avoid normalization, and use the unnormalized lower and upper bounds for relative ranking purpose. We can then use the measure of conflict as

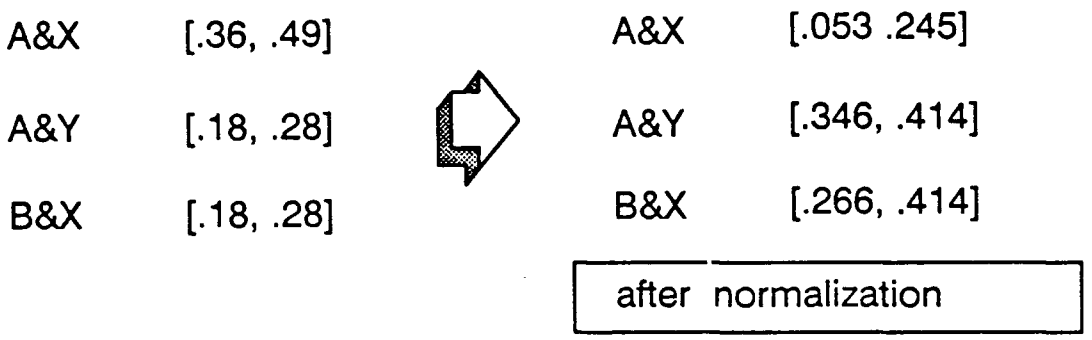


Figure 1.10: Computation of New Bounds (Normalized)

an indicator to determine when to task additional sensors (or to ask the SA Officer) to disambiguate the situation and identify the most reliable sources of information.

1.8 Conclusions

1.8.1 Recommendations for SA - Contact Analysis

We have observed that there is a great pay-off in controlling the generation of scenes generated by the Contact Analysis module. To achieve this goal, we need to attach a *figure of merit* to each scene. This figure of merit can be used to rank the various scenes, to select the most relevant ones (using a dynamic threshold on the figures of merit), to estimate the amount of coverage, and to determine the possible loss of information incurred for not processing the remaining scenes. The figure of merit can be augmented with a *measure of conflict*, indicating the amount of inconsistency among the sources used to define the scene. When this measure of conflict becomes too large, it becomes necessary to identify the sources which must be removed from the information fusion process to maintain a consistent scene. This task can be automatically accomplished either by tasking sensors which can disambiguate the situation, or by generating and comparing possible worlds, in each of which a different input has been eliminated. During this process it is important to have models of the input sources (e.g., acoustics). By knowing the assumptions and other preconditions that determine the applicability of the sensor models, we can identify possible violations of these assumptions/preconditions and determine which information source should be ignored. For example, the presence of a front could invalidate the output of an acoustic model which was designed to handle only normal environmental situations.

Alternatively, this task can be performed interactively, by presenting and explaining the conflicting information and the generated posssible world alternatives to the Situation Assessment Officer.

Since the detection of conflicting information is so important, we should use column 4 in Figure 1.4 to select a reasoning with uncertainty technique capable of recognizing

inconsistencies. From these techniques, our first suggestion is the use of Dempster-Shafer (DS) theory.

Given the computational cost of DS method, however, it is necessary to determine the size of the largest non-decomposable logical dependency graph which is part of the Contact Analysis module.

This functional decomposition is actually required for all three modules in SA: Contact Analysis, Situation Analysis, and Situation Understanding. This analysis must be based on a stable architecture design for SA and its three modules. For each of these modules it is necessary to have the next level design specifications (e.g. for Situation Understanding we need the specifications for Lethality, Intent, Mission Mode and Awareness). These specifications must include the number of *input variables* (and the number of values considered for each variable); the number of *output variables* (and the number of values considered for each variable); a sample of typical functional transformations from inputs to outputs; and a detailed functional thread instantiating a sequence of messages and transformations. The above information can be used to estimate the worst-case computational complexity for Dempster Shafer (and for Bayesian Belief Networks).

If this complexity is still unmanageable, we can always resort to a simplified (and restricted) version of Dempster-Shafer [Bar81], which has time-linear complexity.

1.8.2 Recommendations for SA - Situation Analysis and Understanding

We strongly suggest the use of possibilistic reasoning, as implemented in PRIMO, to deal with the interpretation of the scene. This suggestion is based on three major factors:

1. PRIMO's low computational cost (linear in the number of nodes, under the restriction of unidirectionality, i.e., DAG).

2. PRIMO's natural representation and use of similar, prototypical situations to provide a subjective interpretation/evaluation for a given situation.

3. PRIMO's common semantics with the indices used by the plan retriever module in the Tactical Planner.

1.8.3 Recommendations for TP - Plan Retriever

Based on a preliminary inspection of TP, we have identified the use of a possibilistic reasoning technique to implement a similarity module for TP plan retriever. This retriever can use the high-level output generated by SA (Situation Analysis and Understanding) to generate a set of abstract features which will be compared with the indices stored with the plans.

The following is a sample of indices used by TP Plan retriever:

- Contact Bearing: known
- Contact Distance : close

- Speed of localization: rapid
- Likelihood of O/S/ Detection: low
- *Primary Target Behavior: predictable*
- O/S resources: normal
- Primary target distance: < 20kyd
- CD: stealthy
- Contacts: few(2-3)
- Target geometry: 2 quads
- Ship signature: normal
- Time pressure: moderate

Most of the linguistic values of these indices can be generated by possibilistic rules in SA. Their semantics can be represented by fuzzy numbers on the corresponding universe of discourse (e.g., a *close* contact distance can be defined by a characteristic function showing a fuzzy interval on the units of kiloyards.) Based on this representation it is possible to have a partial pattern matcher between the abstract features describing the situation analysis/understanding and the indices attached to the stored plans. Based on our previous experience in developing a possibilistic similarity module for case retrieval [BBA90], we believe that this representation of uncertainty is the most suitable for the job.

Bibliography

[AR87] A.M. Agogino and A. Rege. Ides: Influence diagram based expert system. *Mathematical Modelling*, 8:227–233, 1987.

[Bar81] J.A. Barnett. Computational methods for a mathematical theory of evidence. In *Proc. 7th. Intern. Joint Conf. on Artificial Intelligence*, Vancouver, British Columbia, Canada, 1981.

[BBA90] P. P. Bonissone, L. Blau, and S. Ayub. Leveraging the Integration of Approximate Reasoning Systems. In *Proceedings of the Third Annual Spring Symposium Series*, pages 1–6. AAAI, 1990.

[BCGS89] Piero P. Bonissone, Dave Cyrluk, John Goodwin, and Jonathan Stillman. Uncertainty and Incompleteness: Breaking the Symmetry of Defeasible Reasoning. In *Proceeding Fifth AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 34–45. AAAI, August 1989.

[BD86] Piero P. Bonissone and Keith S. Decker. Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity. In L. Kanal and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 217–247. North-Holland, Amsterdam, 1986.

[BGD87] Piero P. Bonissone, Stephen Gans, and Keith S. Decker. RUM: A Layered Architecture for Reasoning with Uncertainty. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 891–898. AAAI, August 1987.

[Bon87a] Piero P. Bonissone. Plausible Reasoning: Coping with Uncertainty in Expert Systems. In Stuart Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 854–863. John Wiley and Sons Co., New York, 1987.

[Bon87b] Piero P. Bonissone. Summarizing and Propagating Uncertain Information with Triangular Norms. *International Journal of Approximate Reasoning*, 1(1):71–101, January 1987.

[Bon87c] Piero P. Bonissone. Using T-norm Based Uncertainty Calculi in a Naval Situation Assessment Application. In *Proceedings of the the Third AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 250–261. AAAI, July 1987.

[Bon89] Piero P. Bonissone. Now that I Have a Good Theory of Uncertainty, What Else Do I Need? In *Proceeding Fifth AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 22–33. AAAI, August 1989.

[BS84] B.G. Buchanan and E.H. Shortliffe. *Rule-Based Expert Systems*. Addison-Wesley, Reading, Massachusetts, 1984.

[BT85] Piero P. Bonissone and Richard M. Tong. Editorial: Reasoning with Uncertainty in Expert Systems. *International Journal of Man-Machine Studies*, 22(3):241–250, March 1985.

[BW88] Piero P. Bonissone and Nancy C Wood. Plausible Reasoning in Dynamic Classification Problems. In *Proceedings of the Validation and Testing of Knowledge-Based Systems Workshop*. AAAI, August 1988.

[BW89] Piero P. Bonissone and Nancy C. Wood. T-norm Based Reasoning in Situation Assessment Applications. In L. Kanal, T. Levitt, and J. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 241–256. North-Holland, Amsterdam, 1989.

[CG83] P.R. Cohen and M.R. Grinberg. A Framework for Heuristic Reasoning about Uncertainty. In *Proceedings Eight International Joint Conference on Artificial Intelligence*, pages 355–357. AAAI, August 1983.

[CJJN90] B. Chamberlain, F.V. Jensen, F. Jensen, and T. Nordahl. Analysis in HUGIN of Data Conflict. In *Proc. Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Ma., 1990.

[CK86] Yizong Cheng and Rangasami Kashyap. Irrelevancy of evidence caused by independence assumptions. Technical Report TR-EE 86-17, School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907, 1986.

[Cla84] William J. Clancey. Classification Problem Solving. In *Proceedings of the 3rd National Conference on Artificial Intelligence*, pages 49–55. AAAI, August 1984.

[Coh85] P. Cohen. *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach*. Pittman, Boston, Massachusetts, 1985.

[Dem67] A.P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967.

[DHN76] R.O. Duda, P.E. Hart, and N.J. Nilsson. Subjective Bayesian methods for rule-based inference systems. In *Proc. AFIPS 45*, pages 1075–1082, New York, 1976. AFIPS Press.

[DL87] Edmund H. Durfee and Victor R. Lesser. Planning to meet deadlines in a blackboard-based problem solver. Technical Report COINS-87-07, COINS at University of Massachusetts, 1987.

[Doy83] J. Doyle. Methodological Simplicity in Expert System Construction: The Case of Judgements and Reasoned Assumptions. *The AI Magazine*, 4(2):39–43, 1983.

[DP84] D. Dubois and H. Prade. Criteria Aggregation and Ranking of Alternatives in the Framework of Fuzzy Set Theory. In H.J. Zimmerman, L.A. Zadeh, and B.R. Gaines, editors, *TIMS/Studies in the Management Science, Vol. 20*, pages 209–240. Elsevier Science Publishers, 1984.

[DP85] D. Dubois and H. Prade. Combination and Propagation of Uncertainty with Belief Functions - A Reexamination. In *Proceedings Ninth International Joint Conference on Artificial Intelligence*, pages 111–113. AAAI, August 1985.

[Gil82] R. Giles. Semantics for Fuzzy Reasoning. *International Journal of Man-Machine Studies*, 17(4):401–415, 1982.

[Gin84] M.L. Ginsberg. Non-Monotonic Reasoning Using Dempster's Rule. In *Proceedings Fourth National Conference on Artificial Intelligence*, pages 126–129. AAAI, August 1984.

[GLF81] T.D. Garvey, J.D. Lowrance, and M.A. Fischler. An Inference Technique for Integrating Knowledge from Disparate Sources. In *Proc. 7th. Intern. Joint Conf. on Artificial Intelligence*, Vancouver, British Columbia, Canada, 1981.

[Gly85] C. Glymour. Independence Assumptions and Bayesian Updating. *Journal of Artificial Intelligence*, 25:95–99, 1985.

[Hec86] D. Heckerman. Probabilistic interpretations for MYCIN certainty factors. In L.N. Kanaal and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 167–196. North-Holland, Amsterdam, 1986.

[Hen87] M. Henrion. Practical issues in constructing a bayes' belief network. In *Proc. Third AAAI Workshop on Uncertainty in Artificial Intelligence*, Seattle, Washington, 1987. AAAI.

[HM84] R.A. Howard and J.E. Matheson. Influence diagrams. In R.A. Howard and J.E. Matheson, editors, *The Principles and Applications of Decision Analysis*, volume 2, pages 719–762. Strategic Decisions Group, Menlo Park, California, 1984.

[IFY82] M. Ishizuka, K.S. Fu, and J.P.T. Yao. A rule-based inference with fuzzy set for structural damage assessment. In M. Gupta and E. Sanchez. editors, *Fuzzy Information and Decision Processes*. North Holland, Amsterdam, 1982.

[Ish82] M. Ishizuka. An Extension of Dempster-Shafer Theory to Fuzzy Sets for Constructing Expert Systems. *Seisan-Kenkyu*, 34:312–315, 1982.

[Joh86] R.W. Johnson. Independence and Bayesian Updating Methods. *Journal of Artificial Intelligence*, 29:217–222, 1986.

[KST82] D. Kahneman, P. Slovic, and A. Tversky. *Judgment under Uncertainty: Heuristic and Biases*. Cambridge University Press, Cambridge, England, 1982.

[LD88] S.L. Lauritzen and D.Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Roy. Stat. Soc. Ser. B*, 50, 1988.

[LG83] J. Lowrance and T.D. Garvey. Evidential reasoning: an implementation for multisensor integration. Technical Report Technical Note 307, SRI International, Artificial Intelligence Center, Menlo Park, California, 1983.

[LGS86] J.D. Lowrance, T.D. Garvey, and T.M. Strat. A Framework for Evidential-Reasoning Systems. In *Proc. National Conference on Artificial Intelligence*, pages 896–903, Menlo Park, California, 1986. AAAI.

[LR57] R.D. Luce and H. Raiffa. *Games and Decisions: Introduction and Critical Survey*. John Wiley, New York, 1957.

[Pea82] J. Pearl. Reverend Bayes on Inference Engines: a Distributed Hierarchical Approach. In *Proceedings Second National Conference on Artificial Intelligence*, pages 133–136. AAAI, August 1982.

[Pea85] J. Pearl. How to Do with Probabilities What People Say You Can't. In *Proceedings Second Conference on Artificial Intelligence Applications*, pages 1–12. IEEE, December 1985.

[Pea88a] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, San Mateo, California, 1988.

[Pea88b] Judea Pearl. Evidential Reasoning Under Uncertainty. In Howard E. Shrobe, editor, *Exploring Artificial Intelligence*, pages 381–418. Morgan Kaufmann, San Mateo, CA, 1988.

[Pfa87] Lise M. Pfau. RUMrunner: Real-Time Reasoning with Uncertainty. Master's thesis, Rensselaer Polytechnic Institute, December 1987.

[PZM81] E.D. Pednault, S.W. Zucker, and L.V. Muresan. On the Independence Assumption Underlying Subjective Bayesian Updating. *Journal of Artificial Intelligence*, 16:213–222, 1981.

[Qui83] J.R. Quinlan. Consistency and Plausible Reasoning. In *Proceedings Eight International Joint Conference on Artificial Intelligence*, pages 137–144. AAAI, August 1983.

[Rei80] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Ric83] E. Rich. Default Reasoning as Likelihood Reasoning. In *Proceedings Third National Conference on Artificial Intelligence*, pages 348–351. AAAI, August 1983.

[Rus87] E.H. Ruspini. The logical foundations of evidential reasoning. Technical Note 408, Artificial Intelligence Center, SRI International, Menlo Park, California, 1987.

[Rus89] E.H. Ruspini. On the semantics of fuzzy logic. Technical Note 475, Artificial Intelligence Center, SRI International, Menlo Park, California, 1989.

[Rus90a] E.H. Ruspini. Possibility as similarity: The semantics of fuzzy logic. In *Proceedings 1990 AAAI Uncertainty Workshop*, Cambridge, MA, 1990.

[Rus90b] Enrique Ruspini. The semantics of vague knowledge. *Revue de Systemique*, forthcoming 1990.

[SB75] E.H. Shortliffe and B. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

[SBBG86] L. M. Sweet, P. P. Bonissone, A. L. Brown, and S. Gans. Reasoning with Incomplete and Uncertain Information for Improved Situation Assessment in Pilot's Associate. In *Proceedings of the 12th DARPA Strategic Systems Symposium*. DARPA, October 1986.

[Sch86] R.D. Schachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[SCH90] J. Suermondt, G. Cooper, and D. Heckerman. A Combination of Cutset Conditioning with Clique-Tree Propagation in the Pathfinder System. In *Proc. Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Ma., 1990.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, New Jersey, 1976.

[SK90] P. Smets and R. Kennes. Computational Aspects of the Mobius Transform. In *Proc. Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Ma., 1990.

[Sme81] P. Smets. The Degree of Belief in a Fuzzy Set. *Information Science*, 25:1–19, 1981.

[Sme88] P. Smets. Belief functions. In P. Smets, A. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning*. Academic Press, New York, 1988.

[SS63] B. Schweizer and A. Sklar. Associative Functions and Abstract Semi-Groups. *Publicationes Mathematicae Debrecen*, 10:69–81, 1963.

[SS83] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. North Holland, New York, 1983.

[SS90] P.P. Shenoy and G.R. Shafer. Axioms for Discrete Optimization Using Local Computation. In *Proc. Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Ma., 1990.

[Sti90] J. Stillman. On Heuristics for Finding Loop Cutsets in Multiply-Connected Belief Networks. In *Proc. Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Ma., 1990.

[Str84] T.M. Strat. Continuous belief functions for evidential reasoning. In *Proc. National Conference on Artificial Intelligence*, pages 308–313, Austin, Texas,, 1984.

[Str89] T. Strat. Making Decisions with Belief Functions. In *Proc. Fifth AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 351–360, Windsor, Canada, 1989. AAAI.

[YM86] Halpern J. Y. and Y. Moses. A Guide to Modal Logics of Knowledge and Belief. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 4480–490. AAAI, 1986.

[Zad65] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[Zad75] L.A. Zadeh. Fuzzy logic and approximate reasoning (in memory of Grigor Moisil). *Synthese*, 30:407–428, 1975.

[Zad78] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[Zad79a] L.A. Zadeh. Fuzzy Sets and Information Granularity. In M.M. Gupta, R.K. Ragade, and R.R. Yager, editors, *Advances in Fuzzy Set Theory and Applications*, pages 3–18. Elsevier Science Publishers, 1979.

[Zad79b] L.A. Zadeh. A theory of approximate reasoning. In P. Hayes, D. Michie, and L.I. Mikulich, editors, *Machine Intelligence*, pages 149–194. Halstead Press, New York, 1979.

[Zad83a] L.A. Zadeh. A computational approach to fuzzy quantifiers in natural language. *Computers and Mathematics*, 9:149–184, 1983.

[Zad83b] L.A. Zadeh. Linguistic variables, approximate reasoning, and dispositions. *Medical Informatics*, 8:173–186, 1983.

[Zad84a] L.A. Zadeh. A computational theory of disposition. In *Proc. 1984 Int. Conf. Computational Linguistics*, pages 312–318, 1984.

[Zad84b] L.A. Zadeh. Review of Books: A Mathematical Theory of Evidence. *The AI Magazine*, 5(3):81–83, 1984.

[Zad85a] L.A. Zadeh. A simple view of the dempster-shafer theory of evidence and its implications for the rule of combinations. Technical Report 33, Berkeley Cognitive Science, Institute of Cognitive Science, University of California, Berkeley,, 1985.

[Zad85b] L.A. Zadeh. Syllogistic reasoning in fuzzy logic and its application to usuality and reasoning with dispositions. *IEEE Trans. Systems, Man and Cybernetics*, SMC-15:754–765, 1985.

[Zad88] L.A. Zadeh. Dispositional logic. *Appl. Math. Letters*, 1(1):95–99, 1988.

2. T-norm Based Reasoning in Situation Assessment Applications

Piero P. Bonissone Nancy C. Wood

Artificial Intelligence Program
General Electric Corporate Research and Development
Schenectady, New York 12301
Arpanet: bonissone@ge-crd.arpa woodnc@ge-crd.arpa

Abstract

RUM (Reasoning with Uncertainty Module), is an integrated software tool based on a KEE, a frame system implemented in an object oriented language. RUM's architecture is composed of three layers: *representation*, *inference*, and *control*.

The representation layer is based on frame-like data structures that capture the uncertainty information used in the inference layer and the uncertainty meta-information used in the control layer. The inference layer provides a selection of five T-norm based uncertainty calculi with which to perform the intersection, detachment, union, and pooling of information. The control layer uses the meta-information to select the appropriate calculus for each context and to resolve eventual ignorance or conflict in the information. This layer also provides a context mechanism that allows the system to focus on the relevant portion of the knowledge base, an uncertain-belief revision system that incrementally updates the certainty value of well-formed formulae (*wffs*) in an acyclic directed deduction graph.

RUM has been tested and validated in a sequence of experiments in both naval and aerial situation assessment (SA), consisting of correlating reports and tracks, locating and classifying platforms, and identifying intents and threats. An example of naval situation assessment is illustrated. The testbed environment for developing these experiments has been provided by LOTTA, a symbolic simulator implemented in Flavors. This simulator maintains time-varying situations in a multi-player antagonistic game where players must make decisions in light of uncertain and incomplete data. RUM has been used to assist one of the LOTTA players to perform the SA task.

2.1 Introduction

The trend followed by most approaches for reasoning with uncertainty has shown an almost complete disregard for the fundamental issues of automated reasoning, such as the proper *representation* of information and meta-information, the allowable *inference*

paradigms suitable for the representation, and the efficient *control* of such inferences in an explicitly programmable form. The majority of the approaches to reasoning with uncertainty do not properly cover these issues. Some approaches lack expressiveness in their representation paradigm. Other approaches require unrealistic assumptions to provide uniform combining rules defining the plausible inferences. Most approaches do not even recognize the need for having an explicit control of the inferences.

This lack of awareness has been the driving force for compiling a list of requirements (desiderata) that each reasoning system handling uncertain information should satisfy. Following the typical structure of automated reasoning techniques, the list of requirements has been organized in three layers: *representation*, *inference*, and *control*. The extension of this explicit layered separation from *crisp*-reasoning systems to *uncertain*-reasoning systems is a natural step leading to a better integration of the management of uncertainty with the various techniques for automated reasoning.

An in-depth treatment of the layered desiderata can be found in a previous paper [3]. In this article we describe RUM (Reasoning with Uncertainty Module), which represents our answer to the desiderata. We then illustrate two situation assessment problems which have been used to validate RUM. Both applications are based on an architecture designed to simulate various military scenarios involving Multi-Sensors/Multi-Targets (MS/MT) and to perform situation assessment (SA) related tasks. The MS/MT architecture, illustrated in Figure 2.1, is composed of two major blocks: a reasoning system and a simulation environment.

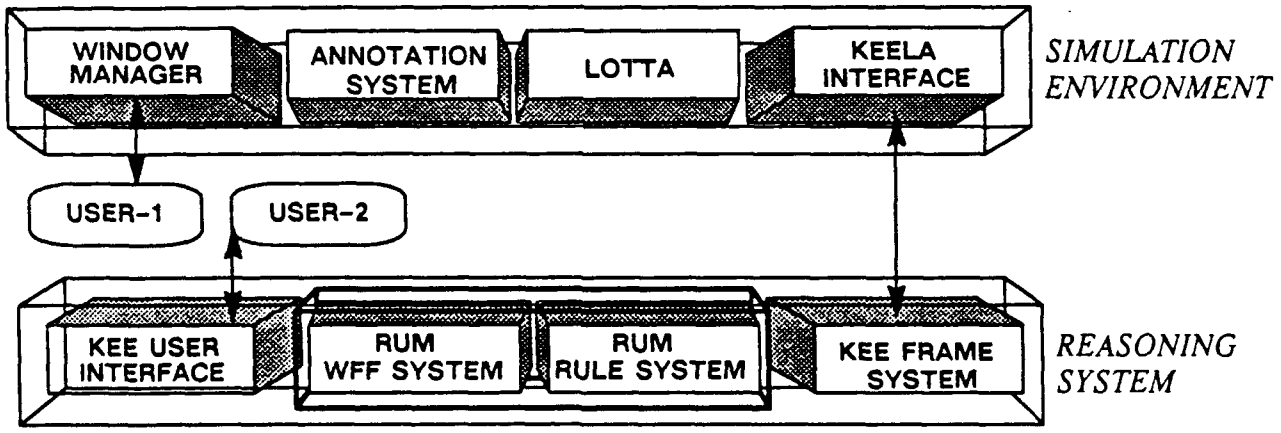


Figure 2.1: Architecture for Multi-Sensors/Multi-Targets (MS/MT)

RUM is the reasoning system used in this architecture. This system, built according to the three layer desiderata, is thoroughly described in [5]. It is summarized in Section 2.2, with a particular focus on its control layer. The second block of the MS/MT architecture, the simulation environment, is described in Section 2.3, in conjunction with some definitions of the tasks required to perform situation assessment. The last two sec-

tions contain an analysis of the MS/MT experiment and some preliminary conclusions on this work.

2.2 RUM, The Reasoning System

RUM is an integrated software tool based on KEE[1], a frame system implemented in an object oriented language [KEE86]. The underlying theory of RUM, centered around the concept of Triangular norms, was described in two previous articles [3], [5]. RUM's architecture is composed of three layers: *representation, inference,* and *control.* A philosophical motivation for RUM's three layer organization can be found in [2]. This section summarizes some of the theoretical results and provides a unified framework for their interpretation and use in RUM's architecture.

2.2.1 Representation: the Wff System and the Rule Language

The representation layer is based on frame-like data structures that capture the uncertainty information used in the inference layer and the uncertainty meta-information used in the control layer.

RUM's Wff System

RUM's Wff System modifies KEE's representation of a *wff* (well-formed formula). RUM's *wff* is the pair [<*unit*> <*slot*>], which is the description of a variable in the problem domain. For each *wff* a corresponding uncertainty unit is created. The unit contains a list of the values that were considered for the *wff*. For each value the unit maintains its certainty's *lower* and *upper bounds*, an *ignorance measure*, a *consistency measure*, and the *evidence source.*

Figure 2.2 illustrates an example of an uncertainty unit attached to a *wff*. The *wff* is the variable [*Platform-439 Class-name*]. In the uncertainty unit, under the slot VALUES, we can see the possible values which were considered by the system and their corresponding certainty bounds. The uncertainty unit also maintains a record of the rule instances which were fired to derive such values (for inferred *wff*s, this logical support represents the evidence source).

RUM's Wff System allows the user to express arbitrary uncertainty granularity by providing the flexibility to mix precise and imprecise measures of certainty in defining the input certainty (points, intervals, fuzzy numbers/intervals, linguistic values) and the rule strengths (categorical and plausible IF/IFF). Various term sets of linguistic probabilities with fuzzy-valued semantics [1] provide a selection of input granularity. The values of the terms can be used as default values or can be modified by the user.

[1]KEE is a trademark of IntelliCorp

RUM's Rule System: The Rule Language

RUM's Rule System replaces KEE Rule System-3 capabilities by incorporating uncertainty information in the inference scheme. The uncertain information is described in the uncertainty units of the *wffs*, represented in RUM's Wff System, and in the degrees of sufficiency and necessity attached to each rule.[2]

The degree of sufficiency denotes the extent to which one should believe in the rule conclusion, if the rule premise is satisfied. The degree of necessity indicates the confidence with which one can negate the conclusion, if the premise fails.

A rule is internally represented by a frame with several slots. These slots include the name of the rule; the lists of contexts, premises, and conclusions; the rule's sufficiency and necessity; and the T-norm to be used for aggregation. All slots (except the name, premises, and consequences) have default values. The contexts, premises, and conclusions can comprise values, variables, RUM predicates and arbitrary LISP functions. Rules with unbound variables are instantiated with the necessary environment to produce rule instances. An example of two RUM rules is provided in Section 2.3.2.

The T-norm specified with each rule is used to aggregate the certainties of the rule premises and to perform detachment (which computes the certainty of the conclusion given the sufficiency and necessity of the rule). It defaults to T_3, which is the MIN function. The associated T-conorm is used to aggregate the certainties of identical conclusions inferred by multiple rule instances derived from the *same* rule. These are often subsumptive, and the value defaults to S_3, the MAX function. Finally, each separate consequence of a rule has a specified T-conorm that will be used to aggregate the consequence with identical consequences derived from *different* rules. (i.e., multiple assignments of the same value to the wff). The negation operator causes the wff to be assigned the complemented value.[3]

2.2.2 Inference: Triangular norms (T-norms) Based Calculi

The inference layer is built on a set of five Triangular norms (T-norms) based calculi. The T-norms' associativity and truth functionality entail problem decomposition and relatively inexpensive belief revision. The theory of T-norms has been covered in previous articles [1], [2], [3], [4], [5]. A brief review of their definition and their use in RUM is included for the reader's convenience.

---

[2]It is important to note that the inference symbol $\rightarrow$ in the production rule $A \xrightarrow{s} B$ is interpreted as a (weak) *material implication* operator in multiple-valued logics. The value s is the lower bound of the degree of sufficiency of the implication. This is in contrast with the interpretation of *conditioning*, i.e., $s = P(B|A)$. The symbol $\leftrightarrow$ in the production rule $A \xleftrightarrow{s,n} B$ is interpreted as a (weak) *logical equivalence* operator in multiple-valued logics, in which s and n are the lower bounds of sufficiency and necessity, respectively. This (weak) logical equivalence is an *if-and-only-if* (IFF) rule, which can be decomposed into the two rules: $A \xrightarrow{s} B$ and $B \xrightarrow{n} A$ (equivalent to $\neg A \xrightarrow{n} \neg B$). RUM's rules are of the type: $C \rightarrow (A \xleftrightarrow{s,n} B)$, where C indicates the context of the rule (see Section 2.2.3) and $\rightarrow$ represents the strong material implication.

[3]If a *wff* has a value A with an If the certainty interval attached to a value A is [L(A), U(A)], its complemented value, $\neg A$, has a certainty interval defined by [1-U(A), 1-L(A)].

Background Information on T-norms

Triangular norms (T-norms) and Triangular conorms (T-conorms) are the most general families of binary functions that satisfy the requirements of the conjunction and disjunction operators, respectively. T-norms and T-conorms are two-place functions from [0,1]x[0,1] to [0,1] that are monotonic, commutative and associative. Their corresponding boundary conditions, i.e., the evaluation of the T-norms and T-conorms at the extremes of the [0,1] interval, satisfy the truth tables of the logical AND and OR operators.

In a previous paper [1], six parametrized families of T-norms and dual T-conorms were discussed and analyzed by the author. Of the six parametrized families, one family was selected due to its complete coverage of the T-norm space and its numerical stability. This family, originally defined by Schweizer & Sklar [6], was denoted by $T_{Sc}(a,b,\mathbf{p})$, where $\mathbf{p}$ is the parameter that spans the space of T-norms. More specifically:

$$\begin{array}{lll} T_{Sc}(a,b,\mathbf{p}) = & (a^{-p}+b^{-p}-1)^{-\frac{1}{p}} & \text{if } (a^{-p}+b^{-p}) \geq 1 \quad \text{when } p<0 \\ T_{Sc}(a,b,\mathbf{p}) = & 0 & \text{if } (a^{-p}+b^{-p}) < 1 \quad \text{when } p<0 \\ T_{Sc}(a,b,\mathbf{0}) = & \lim_{p\to 0} T_{Sc}(a,b,\mathbf{p}) = ab & \text{when } p \to 0 \\ T_{Sc}(a,b,\mathbf{p}) = & (a^{-p}+b^{-p}-1)^{-\frac{1}{p}} & \text{when } p>0 \end{array}$$

Its corresponding T-conorm, denoted by $S_{Sc}(a,b,\mathbf{p})$, was defined as:

$$S_{Sc}(a,b,\mathbf{p}) = 1 - T_{Sc}(1-a,1-b,\mathbf{p})$$

In the same paper it was shown that the use of term sets determines the granularity with which the input certainty is described. This granularity limits the ability to differentiate between two similar calculi; the numerical results obtained by using two calculi whose underlying T-norms are very close in the T-norm space will fall within the same granule in a given term set. Therefore, only a finite, small subset of the infinite number of calculi that can be generated from the parametrized T-norm family produces notably different results. The number of calculi to be considered is a function of the uncertainty granularity.

This result was confirmed by an experiment [1] where eleven different calculi of uncertainty, represented by their corresponding T-norms, were analyzed. To generate the eleven T-norms, the parameter $\mathbf{p}$ in Schweizer's family was given the following values:

$$-1, -0.8, -0.5, -0.3, 0, 0.5, 1, 2, 5, 8, \infty$$

The experiment showed that five equivalence classes were needed to represent (or reasonably approximate) any T-norm, when term sets with at most thirteen elements were used. The corresponding five uncertainty calculi were defined by the common negation operator $N(a)= 1-a$ and the DeMorgan pair $(T_{Sc}(a,b,\mathbf{p}), S_{Sc}(a,b,\mathbf{p}))$ for the following values of p:

$$p = -1 \quad T_1(a,b) = max(0, a+b-1)$$
$$S_1(a,b) = min(1, a+b)$$

$$p = -0.5 \quad T_{Sc}(a,b,-0.5) = max(0, a^{0.5} + b^{0.5} - 1)^2$$
$$S_{Sc}(a,b,-0.5) = 1 - max(0, (1-a)^{0.5} + (1-b)^{0.5} - 1)^2$$

$$p \to 0 \quad T_2(a,b) = ab$$
$$S_2(a,b) = a + b - ab$$

$$p = 1 \quad T_{Sc}(a,b,1) = max(0, a^{-1} + b^{-1} - 1)^{-1}$$
$$S_{Sc}(a,b,1) = 1 - max(0, (1-a)^{-1} + (1-b)^{-1} - 1)^{-1}$$

$$p \to \infty \quad T_3(a,b) = min(a,b)$$
$$S_3(a,b) = max(a,b)$$

RUM's inference layer provides the user with a selection of the five T-norm based calculi described above. They are referred to as $T_1, T_{1.5}, T_2, T_{2.5}, T_3$, respectively.

Operations in a T-norm Based Calculus

For each calculus, four operations are defined in RUM's Rule System: *premise evaluation, conclusion detachment, conclusion aggregation,* and *source consensus.* Each operation in a calculus can be completely defined by a Triangular norm $T(.,.)$, and a negation operator $N(.)$, just as in classical logic any boolean expression can be rewritten in terms of an intersection and complementation operator. A formal justifications for the following definitions can be found in References [3], [5]. The four operations are defined as follows:

Premise evaluation: The premise evaluation operation determines the degree to which all the clauses in the rule premise have been satisfied by the matching *wffs*. Let b_i and B_i indicate the lower and upper bounds of the certainty of condition i in the premise of a given rule. Then the premise certainty range [b,B] is defined as:

$$[b, B] = [T(b_1, b_2, \ldots, b_m), T(B_1, B_2, \ldots, B_m)]$$

Conclusion Detachment: The conclusion detachment operation indicates the certainty with which the conclusion can be asserted, given the strength and appropriateness of the rule. Let s and n be the lower bounds of the degree of *sufficiency* and *necessity*, respectively, of the given rule, and let [b,B] be the computed premise certainty range. Then the range [c,C], indicating the lower and upper bound for the certainty of the conclusion inferred by such rule, is defined as:

$$[c, C] = [T(s,b), S(N(n), B)]$$
$$[T(s,b), N(T(n, N(B)))]$$

The degrees of sufficiency and necessity respectively indicate the amount of certainty with which the rule premise implies its conclusion and vice versa. The sufficiency degree is used with *modus ponens* to provide a lower bound of the conclusion. The necessity degree is used with *modus tollens* to obtain a lower bound for the complement of the conclusion (which can be transformed into an upper bound for the conclusion itself).

Conclusion aggregation: The conclusion aggregation operation determines the consolidated degree to which the conclusion is believed if supported by more than one path in the rule deduction graph, i.e., by more than one rule instance. Each group of deductive paths can have a distinct conclusion aggregation operator associated with it. Let the ranges $[c_j, C_j]$ indicate the certainty lower and upper bounds of the *same* conclusion inferred by m rules instances belonging to the same group. Then, for each group of deductive paths, the range [d,D] of the aggregated conclusion is defined as:

$$\begin{aligned}[d, D] = \; & [S(c_1, c_2, \ldots, c_m), S(C_1, C_2, \ldots, C_m)] \\ & [N(T(N(c_1), N(c_2), \ldots, N(c_m)), T(N(C_1), N(C_2), \ldots, N(C_m)))]\end{aligned}$$

Source Consensus: The source consensus operation reflects the fusion of the certainty measures of the same evidence A provided by different sources. The evidence can be an *observed* fact, or a *deduced* fact. In the former case, the fusion occurs before the evidence is used as an input in the deduction process. In the latter case, the fusion occurs after the evidence has been aggregated by each group of deductive paths. The source consensus operation reduces the ignorance about the certainty of A, by producing an interval that is always smaller or equal to the smallest interval provided by any of the information source. If there is an inconsistency among some of the sources, the resulting certainty intervals will be disjoint, thus introducing a conflict in the aggregated result. Let $[L_1(A), U_1(A)], [L_2(A), U_2(A)], \ldots, [L_n(A), U_n(A)]$ be the certainty lower and upper bounds of the same conclusion provided by n different sources of information. Then, the result $[L_{tot}(A), U_{tot}(A)]$, obtained from *fusing* all the assertions about A, is given by taking the intersection of the certainty intervals:

$$\begin{aligned}[L_{tot}(A), U_{tot}(A)] = \; & [Max_i(L_i(A)), Min_i(U_i(A))] \\ & [S_3(L_i(A)), T_3(U_i(A))]\end{aligned}$$

2.2.3 Control: Calculus selection, Belief Revision, Context Mechanism

Calculi Selection

As it was discussed in the previous section, RUM's Rule System uses a set of five T-norm based calculi. The calculus used by each rule instance is inherited from its rule subclass (the rule before the instantiation). The calculus can be modified through KEE's user interface or programmatically (i.e., by an active value). Class inheritance can also be used to modify the degree of sufficiency and necessity of all the rule members of the same class.

The calculi selection consists of two assignments. The first assignment indicates the T-norm with which the premise evaluation and the conclusion detachment will be

computed. Such an assignment is made for each rule, and, through inheritance, is passed to all rule instances derived from the same rule.

The second assignment indicates the T-conorm (represented by its dual T-norm) with which the conclusion aggregation will be computed. This assignment is made for each subset of rule instances generated from *different* rules and asserting the same conclusion.

Rationale for Calculi Selection

The T-norm characteristics will determine the selection choices. For the first assignment, the T-norm assigned to each rule for the premise evaluation and the conclusion detachment will be a function of the decision maker's *attitude toward risk*. The ordering of the T-norms, which is identical to the ordering of parameter p in the Schweizer & Sklar family of T-norms, reflects the ordering from a conservative attitude ($p = -1$ or T_1) to a non-conservative one ($p \rightarrow \infty$ or T_3). From the definition of the calculi operations, we can see that T_1 will generate the smallest premise evaluation and the weakest conclusion detachment (i.e., the widest uncertainty interval attached to the rule's conclusion). T-norms generated by larger values of p will exhibit less drastic behaviors and will produce nested intervals with their detachment operations. T_3 will generate the largest premise evaluation and the strongest conclusion detachment (the smallest certainty interval).

For the second assignment, the T-norm assigned to the subsets of rule instances (derived from different rules and asserting the same conclusion) will be a function of the *lack or presence of positive/negative correlation* among the rules in each subset. The ordering of the T-norms reflects the transition from the case of extreme negative correlation, i.e., mutual exclusiveness (T_1), through the case of uncorrelation (T_2), to the case of extreme positive correlation, i.e., subsumption (T_3).

Currently, all calculi assignments are explicitly made and modified through the user interface, to exercise the implemented accessing functions. In the next development phase of RUM control layer, the calculi assignments will be made by a set of selection rules expressing the meta-knowledge about the context. These rules will select the T-norms that better reflect the knowledge engineer's desired attitude toward risk and the perceived amount of correlation among the rules used in such a context.

Uncertain-Belief Revision

A daemon-based implementation of the belief revision of the uncertain information is available in the control layer of RUM's Rule System. For any conclusion made by a rule, the belief revision mechanism monitors the changes in the certainty measures of the *wffs* that constitute the conclusion's support or the changes in the calculus used to compute the conclusion certainty measure. Validity flags are inexpensively propagated through the rule deduction graph. Five types of flag values are used:

Good Guarantees the validity of the cached certainty measure detached by the rule instance and aggregated into the associated wff.

Bad (level i) Indicates that the cached certainty measure detached by the rule instance is no longer reliable, since the support of some of the wff's in the premise of this rule instance has changed. The *i*th level indicates the correct order of recomputation.

Inconsistent Indicates that the cached certainty measure associated with the wff is conflicting. The inconsistency can be removed by executing a locally defined procedure (differential diagnosis type of experiment, recency of information, split in possible words with subsets of the original sources, etc.)

Not Applicable Indicates that the context of the rule instance is no longer active and the rule instance contribution to the aggregated certainty measure of the wff should be ignored.

Ignorant Indicates that the cached certainty measure detached by the rule instance is too vague to be useful. The default behavior is to ignore the rule instance contribution to the aggregated certainty measure of the wff. Locally defined procedure could be used to remove the ignorance if so specified.

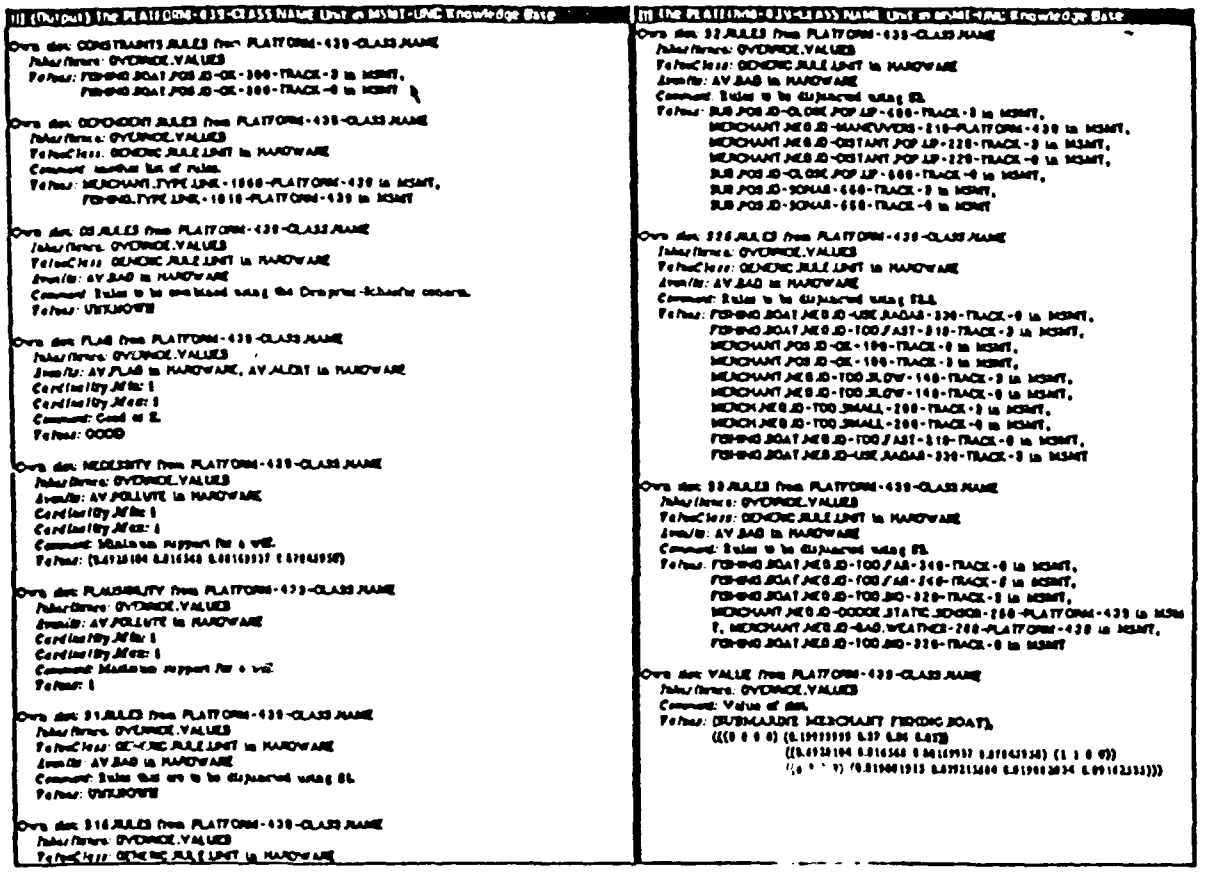


Figure 2.2: Uncertainty Unit Associated with wff [*Platform-439 Class-name*]

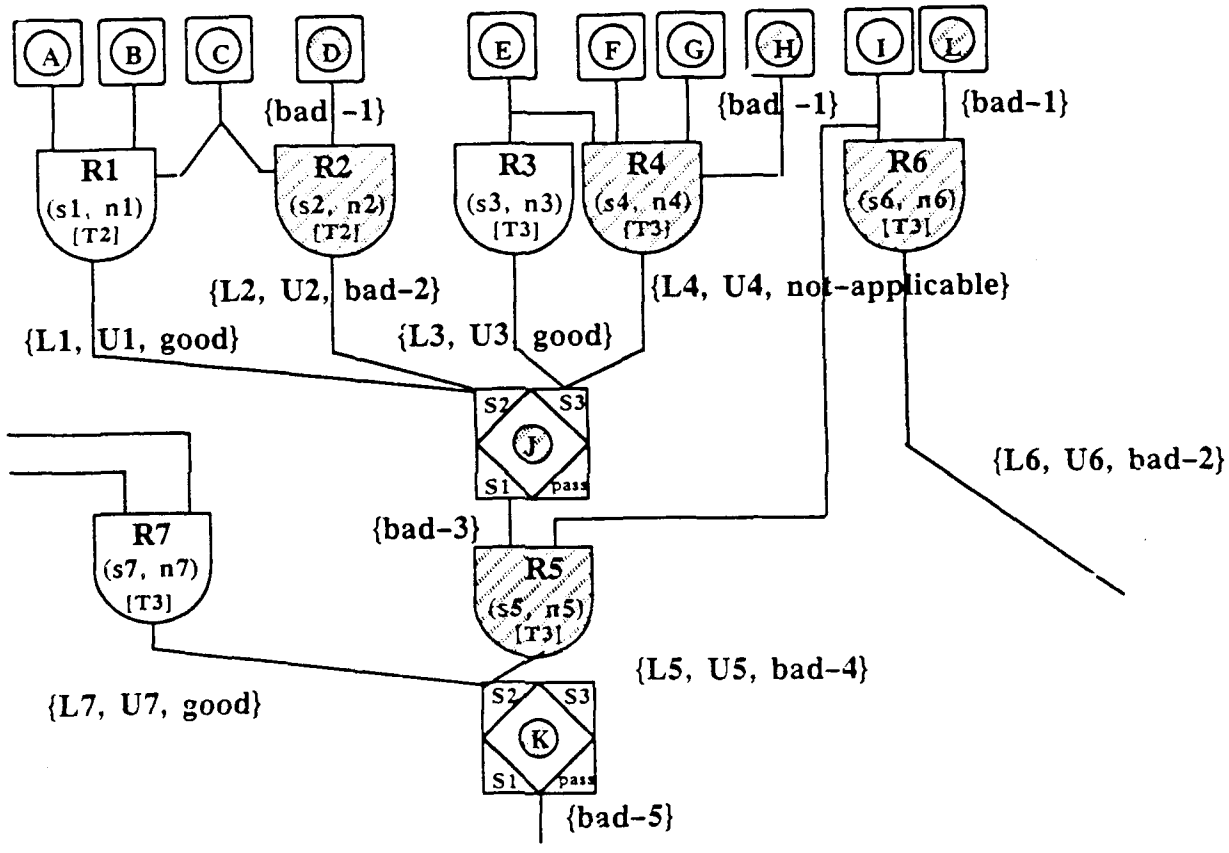


Figure 2.3: Portion of an Acyclic Deductive Graph

An Example of Using the Uncertain-Belief Revision

To provide the reader with a better understanding of the uncertain-belief revision, we will make the following graphical analogy: the *wffs* of the reasoning system correspond to nodes in an acyclic deductive graph; the inference rules in the system correspond to the inference gates that connect the nodes in the graph. There are two types of *wffs*: the observations or assumptions, corresponding to the nodes at the *frontier* of the graph, and the inferred conclusions, corresponding to the *intermediate* nodes in the graph. The first type of node does not have any logical support (its evidence source is the observer or the assumption's maker). The second type of node has a logical support represented by the set of rule instances that made that inference. For this second type, the logical support is the evidence source. Figure 2.3 illustrates a a portion of an acyclic deductive graph, in which seven rule instances are depicted as gates.

In Figure 2.3, C and H (depicted as control lines on the side of a gate) represent two context descriptions that enable/disable the activation of rules R1, R2, R4. The other two rules (R3 and R5) are always potentially active (regardless of context). The figure shows the case in which fact D has just changed. This change causes the propagation of a bad-validity flag that affects the conclusion of rules R2 and R5 (J and K, respectively). The numbers attached to the bad flag indicate the order in which a recomputation of the certainty measures must be performed. Fact H has also changed and its new value no longer satisfies the context description of rule R4, thus causing the not-applicable flag to be attached to the detachment of R4. Fact L has also changed, affecting the validity of Rule R6's detachment.

Reasoning under Pressure

The belief revision system offers both backward and forward processing. Running in *depth-first, backward mode*, RUM recomputes the certainty measures of the modified *wffs* that are required to answer a given query. This mode (called reasoning under pressure) is used when the system or the user decide that they are dealing with time-critical tasks. In the case illustrated in the previous figure, if the value of *wff* K were requested, the systems would perform the following sequence of tasks: fetch the new certainty values of D (lower and upper bounds); recompute the detachment of rule R2; use T-conorm S_2 to evaluate the OR node (with R1 and R2's detachments); ignore R4's detachment, treating R3's detachment as the only input to the OR node associated with T-conorm S_3; fuse the two OR nodes, defining the new certainty values of *wff* J; recompute the detachment of rule R5; use T-conorm S_2 to evaluate the OR node (with R5 and R7's detachments), defining the new certainty values of *wff* K.

When time is not critical, the system can use a *breadth-first, forward mode* processing to recompute the certainty measures of the modified *wffs*, attempting to restore the integrity of the rule deduction graph. In the case illustrated in the previous figure, this implies an update of fact L and rule R6 (both of which were not considered by the back-

ward mode, since they did not play any role in determining the value of the proposed query, e.g. *wff* K).

The structure of the graph can also change, as new rule instances are created or deleted, due to changes in the facts' values, (as opposite to facts' certainty values). The deduction graph is updated and bad flags are propagated throughout the network.

Rule Firing Control via Context Activation

A user-definable threshold can be attached to each rule context, either by local definition or by inheritance from a rule class. A rule context is defined as a conjunction of conditions that must be satisfied before the rule can be considered for premise evaluation. Each condition is described by a predicate on object-level *wffs* (facts in problem domain), or control-level *wffs* (markers asserted by meta-rules). The semantics of a context C attached to an inference rule (establishing the weak logical equivalence between A and B) is given by the following expression:

$$C \rightarrow (A \overset{s,n}{\leftrightarrow} B)$$

where s and n indicate the lower bounds of the degree of sufficiency and necessity that the rule provides; $\rightarrow$ represents the strong material implication; $\leftrightarrow$ denotes the weak logical equivalence.

The context mechanism provides the following features:

- By activating/deactivating subsets of the KB, it limits the number of rules that will be considered relevant at any given time, thus increasing the overall system efficiency.
- By only considering the rules relevant to a given situation, it allows the knowledge engineer to effectively use the necessary conditions in the rule's premise. It is now possible to distinguish between the failure of a necessary test (described in the premise) and the failure of the rule's applicability (traditionally described by other clauses in the same premise and now explicitly represented in the context).
- By using predicates on the control-level *wffs*, it provides the required programmability for defining flexible control strategies, such as causing sequences of rules to be executed, firing default rules, ordering and handling time-dependent information, etc.
- By using hierarchical contexts, it can be used as an organizing principle for the knowledge acquisition task.

2.3 The Object Based Simulation Environment

The second block of the MS/MT architecture is the simulation environment. This environment is centered around LOTTA, an object-oriented symbolic battle management sim-

ulator that maintains time-varying situations in a multi-player antagonistic game [BA88]. The development environment based on LOTTA constitutes a testbed for validating new techniques in reasoning with uncertainty and for performing information fusion functions [SBBG86]. The development environment is composed of four basic modules: the *window manager*, the *annotation system*, the *symbolic simulator (LOTTA)*, and the *Interface (KEELA)* . The simulation environment was used to program both naval and aerial scenarios, in which the information fusion and situation assessment tasks were performed.

2.3.1 The Information Fusion/Situation Assessment Problem

The Information Fusion (IF)/Situation Assessment (SA) requires a variety of tasks in which uncertainty pervades both the input data and the knowledge bases. Beside its intrinsic uncertainty, usually the information dealt in each task is also incomplete, time-varying, and, sometimes, erroneous. Thus, the SA problem represents a strong challenge for most automated reasoning systems, since it requires an integration of the uncertainty management with a truth maintenance system (belief revision system) to maintain the integrity of the inference base (or of its relevant subset). The SA problem also requires the reasoning system to detect useless and contradicting information, rejecting the former and resolving the latter.

There is no uniformly agreed definition of what a situation assessment problem entails. The following definitions have been compiled and summarized from a variety of sources [Cla81], [LGFF84] to succinctly describe the SA problem. Given a platform (aircraft, ship, tank) in a potentially hostile environment, the process of performing Situation Assessment consists of the following tasks:

1. Sensor data must be collected from various sources and described as reports.
2. Time-stamped sensor reports must be consolidated into tracks (each track is the trace of an object followed by a given sensor).
3. Tracks associated to the same object must be fused into a platform.
4. The detected platform must be classified and identified (by class and type).
5. Node organization (formation of the identified platforms), use of special equipment, and maneuvering must be recognized.
6. Using the knowledge of the opponent's doctrines and rules of engagement, the recognized formation and observed use of special equipment must be explained by a probable intent, which is then translated into a threat assessment (*retrospective SA*).
7. This analysis is then projected into the future to evaluate plausible plans and to determine likely interesting developments of the current situation (*prospective SA*).

The first four tasks (1-4) define what is generally known as Information Fusion and (low-level) Situation Assessment problems. These tasks determine the scope of the first SA experiment. The last four tasks (4-7) define the Situation Assessment problem and are illustrated in the second SA experiment.

2.3.2 Example of RUM rules

The RUM knowledge base (KB) used in MS/MT application is composed of approximately forty rules, each of which can be instantiated by new sensor reports, new tracks, or new platforms. A representative sample of such a KB is provided by the following two rules.

English Version of Rule-500 (identifying submarines):

Assuming that a radar was used to generate a sensor report (that with other reports generated by the same sensor has been attached to a track associated with a platform), if the first time that the platform was detected (in the track's first report), the platform was located at a distance of at most twenty miles from our radar (i.e., it was a close-distance radar pop-up) then it is most likely that the platform is a submarine. Otherwise, there is a small chance that it is not a submarine.

RUM's Version of the same rule:

```
(add-template 'sub.pos.id-close.pop.up-500 ; Name
   'msmt ; KB
   '((u-lessp (get.uncertain.value (get.value ?track 'first.report) 'range
      (fuzz 20))) ; Premise-list
   '(((get.value ?track 'platform) class.name submarine s2.rules))
; Consequence-list
   '((?track first.report))    ; List of wffs in premise
   '(?track) ; List of units in premise
   '((is-in-class? (get.value ?report 'track) 'source '(radar lotta)))
; Context
   '(most.likely small.chance) ; Sufficiency and necessity
   't3 ; Aggregation T-norm
   '(submarine track.templates)) ; Rule class & instantiation templ.
```

English Version of Rule-550 (identifying submarines):

Assuming that a sonar was used to generate a sensor report (that with other reports generated by the same sensor has been attached to a track associated with a platform), if the detected platform has a low noise emission, and is located at a depth of at least twenty meters, then it is extremely likely that it is a submarine. Otherwise, it may not be a submarine.

RUM's Version of the same rule:

```
(add-template 'sub.pos.id-sonar-550 ; Name
   'msmt ; KB
   '((is-value?  ?report 'noise-emissions 'low) ; Premise-list
     (u-lessp (get.uncertain.value ?report 'elevation) (fuzz -
20)))
   '(((get.platform ?report) class.name submarine s2.rules)) ; Consequence
list
   '((?report elevation)) ; List of wffs in premise
   '(?report) ; List of units in premise
   '((is-in-class? (get.value ?report 'track) 'source '(sonar lotta)))
; Context
   '(extremely.likely it.may) ; Sufficiency and necessity
   't3 ; Aggregation T-norm
   '(submarine report.templates)) ; Rule class & instantiation templ.
```

Notes on the Calculi Selection for Rule 500 and 550

The T-norm used to detach the conclusion of rule 500 and 550 is T_3. This is due to the fact that we want to obtain the smallest certainty interval associated with the detached conclusion. The T-conorm used to aggregate the certainties of the detachments of both rules is S_2. This assignment indicates a lack of correlation among the two rules, which is substantiated by the fact that independent sources of information (radar and sonar) are used in the context of the two rules.

2.4 Experiments in Situation Assessment

2.4.1 Information Fusion and Platform Typing in a Naval Scenario

The first experiment dealt with a naval scenario and has been reported in [4], [BW88]. The experiment was a modified version of the naval situation assessment scenario used by Naval Ocean System Command to test STAMMER [BM79] and STAMMER2 [MMK79], [Fer81]. In this modified scenario, a CGN-36 missile cruiser operating a passive sensor and an SPS-10 surface radar faces two unknown platforms. One of the two platforms

(selected from a large set of ships) is using an active sensor (navigational radar), while the second platform is not using any active sensor.

The cruiser's task was to track, correlate, and classify each detected object. The passive and active sensors were turned on, generating sensor reports which were translated through the KEELA interface into observed *wffs*. The information returned by the passive sensor (GPS-3) contained the heading, position, range, speed, and time at which the platform was detected. This information was attached to a track (TRACK-10), which maintained subsequent sensor reports generated by the same sensor and associated with the same platform (PLATFORM-439). A second track (TRACK-3) for the platform was similarly generated by the SPS-10 radar. A third track (TRACK-7), also generated by the cruiser's active sensor, was generated for the second platform. Figure 2.4 illustrates a portion of the knowledge base where the report, track, and platform information is stored. In the same figure it is possible to observe the rule instantiation (by track) of the two rules (500 and 550) described in Section 2.3.2.

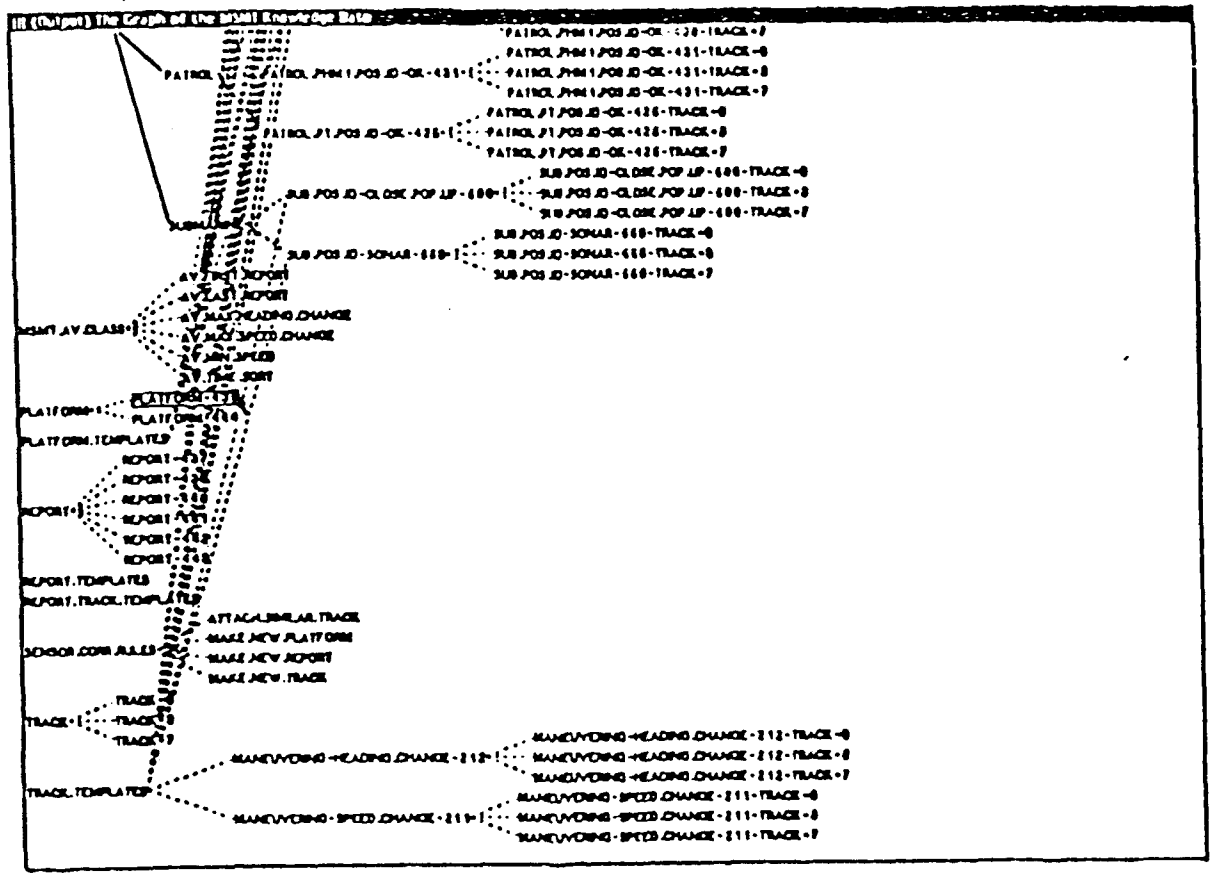


Figure 2.4: Subgraph of the MSMT Knowledge Base

The query posed to RUM was to deduce the class value of the first platform from the tracks information. Three values for the platform class were considered by the system

and qualified by their corresponding certainty bounds: *Merchant* [0.69 1], *Submarine* [0 0.2], and *Fishing Boat* [0 0.02], *Merchant* being best because of the ranking of certainty measures. The lower bound of 0.69 indicates a large amount of positive (confirming) evidence. The upper bound of 1.0 indicates the absence of any negative (refuting) evidence. The class *Submarine* obtained no confirming evidence and a large amount of negative evidence. The refuting evidence was provided by a rule which from the failure to observe a close-distance radar pop-up determined that there was only a small chance for the platform to be a submarine. The class *Fishing Boat* also had no confirming evidence and an overwhelming amount of negative evidence. This refuting evidence was due to the fact that the platform was too far from the fishing areas, too big for a fishing boat, and was using a radar (rules 340, 320, and 330). This information can be obtained from Figure 2.2, by observing the logical support for each of the three value assignments considered for the *wff* [*Platform-439 Class-name*], and from Figure 2.5, by observing the dominant rules for each value. Each rule instance, fired to infer a value of the *wff*, has a cached certainty value (lower and upper bounds) and an associated validity flag. Thus, Figure 2.5 provides the information which was schematically described by the acyclic graph depicted in Figure 2.3.

2.4.2 Tactical Aerial Situation Assessment

The second experiment dealt with tactical aerial situation assessment. The purpose of the experiment was to provide a fighter pilot with the intent evaluation of various potential threats. The simulator generated a variety of scenarios in which up to three aircraft exhibited sufficiently interesting behavior (flight paths intercepting/converging toward ownship, specific sensor use, etc.) to justify a closer analysis. RUM deduced the aircraft's intent from a variety of factors. First the aircraft's class and type was identified by a set of rules based on behavioral information. This inference determined characteristics such as a likely weapon configuration, a likely sensor configuration and an estimate of the Launch Acceptability Region (LAR). Intent was then determined by a second set of RUM rules, based on aspect angle, change in aspect angle, velocity, acceleration, radar mode, ownship detectability template (ODT), shortest time to threat's LAR, and formation. In this experiment, the reasoning system correctly evaluated various intent values chosen among *engage-now, engage-later, influence, evade,* and *non-reactive.* Each plausible intent value was qualified by an uncertainty measure and, from the induced partial ordering, the most likely intent was returned.

2.5 Remarks and Conclusions

RUM's layered architecture properly addresses the requirements imposed by the SA problem. The representation layer captures the uncertain information about the *wffs* (lower and upper bounds) used by the calculi in the inference layer to determine the uncertainty of the conclusions. The representation layer also captures the uncertain meta-

information (evidence source or logical support, measures of ignorance and conflict) used by the belief revision system and other mechanisms in the control layer.

The inference layer provides the knowledge engineer with a rich selection of well-understood calculi to properly represent existing correlations among rules. Numerical computations performed in this layer are efficiently implemented by using a four parameter representation for the uncertainty bounds, supported by a set of closed form formulae that implement the truth functional uncertainty calculi [1].

The control layer provides the explicit selection and modification of uncertainty calculi. Its context activation mechanism allows the reasoning system to focus on the relevant subsets of the changing inference base (the acyclic deductive graph). The uncertain-belief revision maintains the integrity of those relevant subsets, reflecting the changes of the information. RUM's development environment provides the traceability of *wffs* and rules that is required for proper KB development and refinment.

The MS/MT experiment described in this paper has been used to illustrate RUM's capabilities in an IF/SA application. It is a complete experiment, but certainly not a complex one. A more strenuous and realistic validation of RUM is in progress: currently RUM is successfully being used as the reasoning system of the Situation Assessment module in DARPA's Pilot's Associate Program [SBBG86]. In this application, the six tasks (described in Section 2.3.1) that comprise the retrospective SA problem are addressed by RUM in Scenarios involving up to twenty platforms. This application is also used to derive some of the real-time requirements that will represent the focus of RUM's future development.

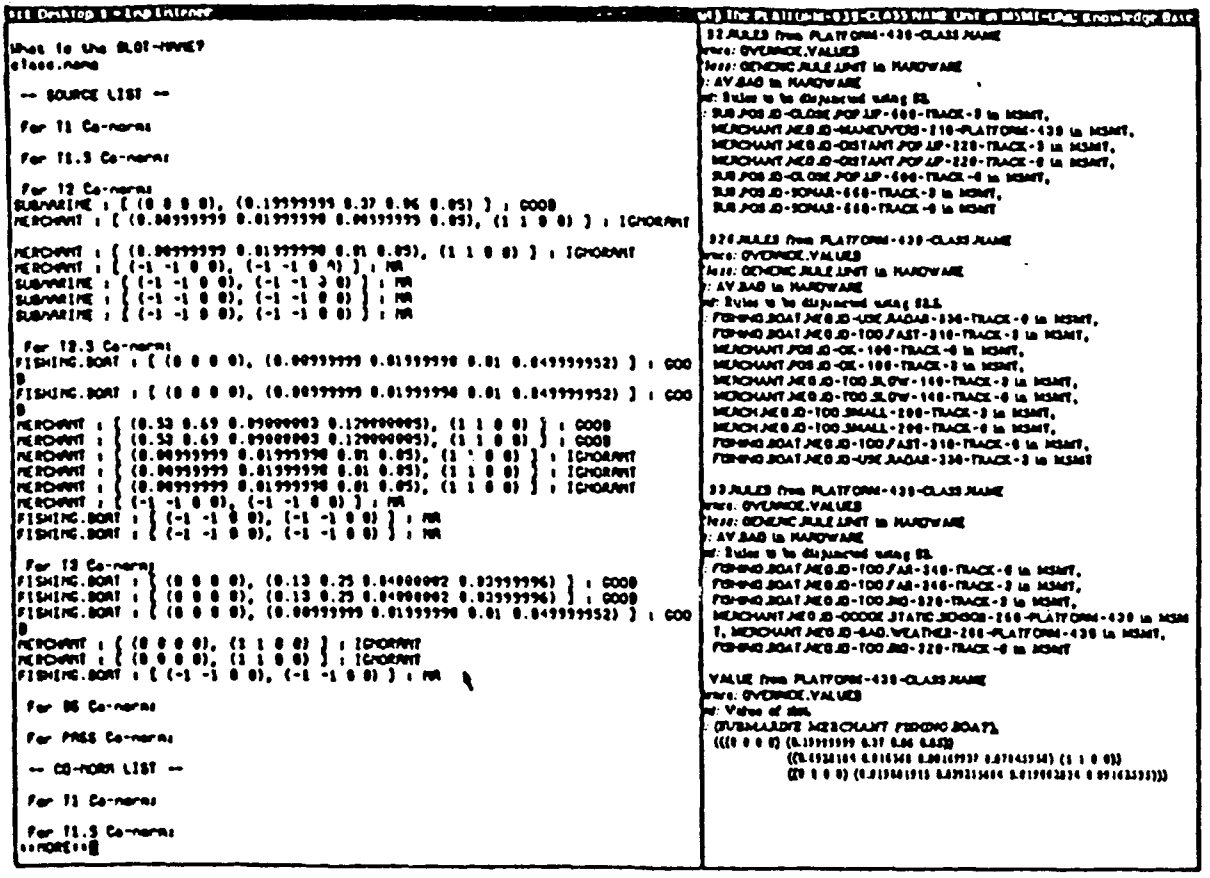


Figure 2.5: Relevant Rule Instances in [*Platform-439 Class-name*] Logical Support

Bibliography

[BA88] Piero P. Bonissone and James K. Aragones. LOTTA: An Object Based Simulator for Reasoning in Antagonistic Situations. In *Proceedings 1988 Summer Computer Simulation Conference*, pages 674–680, SCS, July 1988.

[BD86] Piero P. Bonissone and Keith S. Decker. Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity. In L. Kanal and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 217–247, North-Holland, Amsterdam, 1986.

[BM79] R. J. Bechtel and P. H. Morris. *STAMMER: System for Tactical Assessment of Multisource Messages, Even Radar*. Technical Report 252 (NTIS ADA116527), Naval Ocean System Command, S.Diego, CA, May 1979.

[BW88] Piero P. Bonissone and Nancy C. Wood. Plausible Reasoning in Dynamic Classification Problems. In *Proceedings 1988 Validation and Testing of Knowledge-Based Systems Workshop*, AAAI, August 1988.

[Bon87a] Piero P. Bonissone. Plausible Reasoning: Coping with Uncertainty in Expert Systems. In Stuart Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 854–863, John Wiley and Sons Co., New York, 1987.

[Bon87b] Piero P. Bonissone. Summarizing and Propagating Uncertain Information with Triangular Norms. *International Journal of Approximate Reasoning*, 1(1):71–101, January 1987.

[Bon87c] Piero P. Bonissone. Using T-norm Based Uncertainty Calculi in a Naval Situation Assessment Application. In *Proceedings of the the Third AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 250–261, AAAI, July 1987.

[Bon87d] Piero P. Bonissone, Stephen Gans, and Keith S. Decker. RUM: A Layered Architecture for Reasoning with Uncertainty. In *Proceedings 10th International Joint Conference on Artificial Intelligence*, pages 891–898, AAAI, August 1987.

[Cla81] A. Clarkson. *Toward Effective Strategic Analysis: New Applications of Information Technology*. Westview Press, Boulder, Colorado, first edition, 1981.

[Fer81] J. P. Ferranti. *Evaluation of the Artificial Intelligence Program STAMMER2 in the Tactical Situation Assessment Problem*. Master's thesis, Naval Postgraduate School, Monterey, CA, March 1981.

[KEE86] *KEE Software Development System User's Manual*. Intellicorp, 1986. Version 3.0.

[LGFF84] T. S. Levitt, G. J. Courand, M. R. Fehling, R.M. Fung, C.F. Kaun, and R.M. Tong. *Intelligence Data Analysis*. Technical Report TR-1056-6-Volume 1, Advanced Decision Systems, Mountain View, CA., 1984.

[MMKB79] D. C. McCall, P. H. Morris, D.F. Kilber, and R. J. Bechtel. *STAMMER2 Production System for Tactical Situation Assessment, Vol 1,2*. Technical Report 298 (NTIS ADA084038 ADA084053), Naval Ocean System Command, S.Diego, CA, October 1979.

[SBBG86] L. M. Sweet, P. P. Bonissone, A. L. Brown, and S. Gans. Reasoning with Incomplete and Uncertain Information for Improved Situation Assessment in Pilot's Associate. In *Proceedings of the 12th DARPA Strategic Systems Symposium*, DARPA, October 1986.

[SS63] B. Schweizer and A. Sklar. Associative Functions and Abstract Semi-Groups. *Publicationes Mathematicae Debrecen*, 10:69–81, 1963.

3. Plausible Reasoning in Dynamic Classification Problems

Piero P. Bonissone Nancy C. Wood

Artificial Intelligence Program
General Electric Corporate Research and Development
Schenectady, New York 12301
Arpanet: bonissone@ge-crd.arpa woodnc@ge-crd.arpa

Abstract

The development of reasoning systems addressing situation assessment problems presents a major difficulty common to the development of all expert systems: testing and validating the knowledge base and inference techniques. To solve this problem, and to address a broader class of problems, referred to as *dynamic classification problems*, we have implemented a software architecture capable of generating, interpreting, and resolving complex time-varying scenarios. The test-bed architecture is composed of two parts: a simulation environment, *LOTTA*, and a reasoning system, *RUM*.

The simulation environment is composed of four basic modules: the *window subsystem*, a window based user interface for displaying maps; the *annotation subsystem*, an intelligent database for displaying time varying features; *LOTTA*, the simulator; and a set of tools for interfacing to a reasoning system. LOTTA is a symbolic simulator implemented in an object-oriented language (Symbolics Flavors). LOTTA maintains time varying situations in a multiple player antagonistic game where players assess situations and make decisions in light of uncertain and incomplete data. LOTTA has no reasoning capabilities; these are provided by external reasoning modules, easily interfaced to the LOTTA data structures.

RUM [5], a development environment for reasoning with uncertainty, and RUMrunner, RUM's run-time counterpart, are the reasoning systems used in the test-bed architecture. Both RUM and RUMrunner are based on the theory of plausible reasoning [2], developed at GE CR&D over the last three years. RUM's main function is to build rule-based reasoning systems following the rapid prototyping methodology. Following the testing, and verification of the application using RUM, the knowledge base generated by RUM is then automatically translated and compiled into compact data structures. RUMrunner reasons opportunistically with these data structures to achieve the run-time performance required by most real-time applications.

This software architecture has been used to solve two examples of situation assessment problems. In a naval scenario, it has been tested in information fusion tasks, such as track correlation and platform typing. In an aerial scenario, it has

been used to determine the threat class, type, intent, opportunities, and capabilities of targets.

3.1 Dynamic Classification Problems

The classification problem consists of recognizing a situation from a collection of data and selecting the best action in accordance with some objectives. Examples of classification include diagnosing faulty components, modeling users in terms of goals and beliefs, selecting components from a catalog of items in order to meet certain requirements, performing theoretical analysis, and developing skeletal plans. The classification problem has a recurrent solution structure, as was observed by Clancey [Cla84]. A collection of data, generated from several sources, is interpreted as a predefined pattern. The recognized pattern is mapped into a set of possible solutions, from which one is selected as the most appropriate for the given case. This process is considered a *static* classification problem, since the data are assumed to be invariant over time or at least invariant over the time required to obtain the solution.

A more challenging classification problem is the one in which the environment from which data are collected changes at a rate comparable with the time required to obtain a refined solution. Examples of such *dynamic* classification problems are real-time situation assessment (e.g., air traffic control), real-time process diagnosis (e.g., airborne aircraft engine diagnosis), real-time planning, and real-time catalog selection (e.g., investment selection during market fluctuations). The characteristic structure of this class of dynamic classification problems is illustrated in Figure 3.1.

Situation assessment (SA) [SBBG86], as part of the more extensive battlefield management problem, is a prototypical case of the dynamic classification problem. The retrospective component of situation assessment consists of analyzing and associating observed events to identify and understand those which are relevant. The prospective component consists of projecting the relevant events and assessing their future impact. The correct assessment of current and future impacts of a given situation requires continuous classification in a dynamic environment.

The development of reasoning systems addressing dynamic classification problems presents another difficulty: testing and validating the knowledge base and inference techniques [BB85]. For the static classification problems, such as troubleshooting, this is a relatively simple task: the reduced complexity of the problem domain allows the expert easy generation of test cases. Final verification can be obtained by operating the expert system in the field, solving actual problems. For the battlefield management case, the complexity of the problem domain does not allow the expert to create test cases manually and no actual cases are generally available for testing the expert system in the field.

To solve this problem, and to address a broad class of dynamic classification problems, we have implemented a software architecture capable of generating, interpreting, and resolving complex time-varying scenarios.

This paper describes the test-bed architecture (Section 3.2) and the simulation environment (Section 3.3), then proceeds to discuss the reasoning system (Section 3.4), illustrates two examples of tactical situation assessment implemented in the proposed test-bed architecture (Section 3.5), and concludes with an illustrated description of the methodology used to test and validate the knowledge bases (Section 3.6).

3.2 Test-bed Architecture

The software architecture is composed of two major modules: a *simulation environment*, capable of maintaining the dynamic states of numerous simulated objects; and a *reasoning system*, capable of dealing with the uncertain, incomplete, and time-varying information. This software architecture is illustrated in Figure 3.2.

3.3 Simulation Environment

Large scale simulators have been traditionally implemented in oversized, monolithic Fortran programs. Usually, these simulators perform number-crunching computations to determine the numerical value of every available simulation parameter. However, it has been noticed by McArthur [MKN86] that these traditional simulators are too restrictive. They do not provide the selective richness and flexibility required to exercise and validate the broad gamut of reasoning tasks required to solve dynamic classification problems. Four major shortcomings have been identified by McArthur [MKN86]: the inability to verify the completeness and accuracy of the models; the inability to modify models and construct alternative models; the incomprehensibility of the results; and the long required run times. This view lead the RAND group to the development of ROSS [MKN86] as the underlying object-oriented simulation language used to implement a variety of applications, such as SWIRL [KMN82] and TWIRL [KEGN86]. Recently, an object-oriented based simulator has been used to provide enough complexity and uncertainty in the generated problem space to create challenging situations for a mobile robot planning system [FH87].

We have adopted the object-oriented methodology to implement LOTTA, a symbolic simulator, which, upon demand, can provide numerical information. Due to evolving requirements, LOTTA has undergone a large number of iterative refinements, as described by the rapid prototyping paradigm [Pre87]. As new scenarios were generated by LOTTA, new objects had to be defined, new features had to be displayed, and more accurate sensor/weapon models had to be included. The object-oriented language used to implement LOTTA has been essential in providing the modeler with the requisite flexibility.

The simulation environment is composed of four basic modules: the *window subsystem*, a window based user interface for displaying maps; the *annotation subsystem*, an intelligent database for displaying time varying features; *LOTTA*, the simulator; and a set of tools for interfacing to a reasoning system.

3.3.1 Window Subsystem

The window subsystem is a window based user interface for displaying maps. It controls the menu-driven interaction of the human player with LOTTA and handles multiple windows per player. By interacting with the window subsystem, each player can create, inspect, or delete *objects*; set up, execute, display *orders* for each object; zoom in and out of the *display map*; create or kill new *windows*; create, inspect, modify, rename, delete, or display *features*; and automatically run test cases, among other things.

3.3.2 Annotation Subsystem

The annotation subsystem is an intelligent database for LOTTA. It is composed of a feature extraction system and a feature watcher. The feature extraction system allows both simple and complex time-varying features to be calculated and stored along with the dependencies and recalculate function which allow the feature to be maintained over time. Every feature, whether internally or externally computed, has multiple views (graphical representations for use in decision-making and explanation tasks). The feature watcher maintains the dependency directed information that characterizes the dynamic support of the features and monitors the support for possible changes. The watcher will then guide the "lazy" recomputation of those features whose support has changed since the feature's last computation.

Some of the features that can be created for objects are: *parameters* (e.g., size, maximum speed), ranges (e.g., weapon or movement), movement orders (e.g., path, velocity, altitude profiles), sensor orders (e.g., types and modes), piece data (e.g., altitude, speed, heading), image data (e.g., bearing, range, altitude, speed, heading), detection (e.g., sensor ranges and probability of detection), and Launch Acceptability Region (LAR). Some of the views available for these feature types include splines, vector fields, numeric or textual annotations, and field contours.

3.3.3 Simulator

The core of the simulation environment is the symbolic, object-oriented simulator. This simulator maintains time-varying situations in a multi-player antagonistic game where players must make decisions in light of uncertain and incomplete data. The structure of this simulator is similar to that of multiple player antagonistic games in which each player has only partial information.[1] Note that a *separate* simulator exists for each player, preventing unauthorized information usage, but necessitating a robust communications scheme.

The simulator maintains a world model, composed of *static* and *dynamic* elements whose states change as a result of the decisions and actions made by each player. The

[1] Each player's knowledge about the opponent's assets is obtained by the simulated use of its sensors. Under the default assumption, each player has perfect information about its own pieces. This assumption can be easily removed by forcing the player to rely only on information acquired through its sensors.

static knowledge includes inalterable a priori information such as terrain topology, terrain type, locations of fixed obstacles (impassable for navigational purposes), organizational structures and sizes of the teams of agents which will play an active role in the simulation, etc. The dynamic knowledge describes time-varying information such as weather (which can be partially predicted, but whose behavior cannot be influenced), removable obstacles (such as bridges and land mines), as well as friendly and unfriendly objects that move, observe, and act in this micro-world according to their associated orders.

Following the design philosophy of structured programming, LOTTA was built in a modular fashion. The playing pieces, the players, and the simulation control flow are all implemented as distinct modules.

Playing Pieces

In LOTTA *all* the elements of the simulation are defined as Flavors instances, (i.e., objects with multiple inheritance). Message passing is the uniform communication paradigm used for sending commands and modifying the internal states of the objects. In a traditional (not object-oriented) structured programming paradigm, the approach suggests a separate data entity for each playing piece on the board. A set of subroutines would be available for each class, but as the number of subroutines and classes increase, naming problems for *different but similar* actions arise. For example, planes, submarines and trucks all may move (i.e., change their location on the map), but their movements are constrained by different media, conditions, capabilities, and obstacles. A plane may fly through the air or taxi down a runway, a submarine may move underwater or on the surface, and a truck may move along paved roads, all assuming no collisions, favorable weather conditions, etc.

By defining playing pieces as objects with multiple inheritance, the complexity of the above situation can be significantly decreased. By providing each piece with movement capabilities described by a *mixin*, the common need to change locations on the map is shared by all the pieces. More specific movement capabilities can then be defined for each class. For instance, the underwater capabilities of a submarine would be described by a *submarine-movement-mixin* that specializes the more generic *ship-movement-mixin*, which in turn is built upon the most basic *movement-mixin* flavor.

In LOTTA's implementation, the family of movement flavors handles more than just coordinate changes. Other operations, such as fuel consumption and collision avoidance, must be included under the broad heading of movement. For instance, for those pieces whose movements require the expenditure of fuel, a refueling mechanism must also be provided. The accounting of fuel has been written as yet another *mixin* and is used in conjunction with the *movement-mixin* flavor.

Each playing piece (i.e., each dynamic element in the simulation), is built from many simpler component flavors, from which it inherits various methods for processing incoming messages regarding weapons, sensors, damage, repairs, movements, transportation, etc. By combining these flavors with different parameters, many different types of playing pieces have been created.

LOTTA's Control Flow

The control flow of the simulation is obtained by creating a main loop (the simulation cycle), which coordinates communication between the players. The simulation cycle is divided into 8 phases: *GAME-SYNC, SENSOR (PROBE & ECM), MOVEMENT, SENSOR (PROBE & ECM), COMBAT (CIDS and OFFENSE), and MOVEMENT.* Figure 3.3 illustrates these phases.

At the end of each movement phase, time is incremented by half of the real-time value assigned to the cycle. The underlying assumption is that the time required for weapon and sensor allocation is insignificant when compared with the time required to move. By dividing the simulation phase into the above eight phases, each player may assess the current state before committing to movement or weapon allocation decisions.

Each player can only give orders to its pieces (i.e., send messages) related to the current phase in the cycle. When a player has finished giving orders for the current phase, the orders are executed and it waits for the other players to complete their corresponding phases. The control flow then advances to the next phase and the process is repeated. Upon receiving an order, each piece attempts to execute it; in normal operation, these orders are completed during this phase. Some orders (such as turning off a sensor) can change the object's internal state and maintain it until new orders arrive. Other orders (such as move to a given location) are removed from the object's order list as they are executed. By the end of a phase, any number of actions may have been completed by each playing piece. The decision maker may assign these orders programmatically, by menu selection, or by direct editing of the data structures with a specially provided tool.

This protocol is necessary to maintain a breadth-first propagation of messages through the network and prevent the results of the simulation from depending on the order in which each piece received the messages. This synchronization is also essential in distributing the decision making capabilities throughout clusters of pieces in the network.

3.3.4 Interface

Since the purpose of the LOTTA simulation system is to provide an environment for testing expert systems, a mechanism for transferring the states of the simulated objects to the reasoning system is required. In addition, this mechanism must continually inform the reasoning system of changes in the simulation. Additional flexibility is gained by allowing the reasoning system the ability to send orders to pieces. In manual mode, the player attaches commands to each piece which are examined and executed before the next phase. By replacing one of the players with an inference system, it is possible to test the inference system interactively.

Provisions could be made to allow a switch from the currently implemented *centralized* decision maker, to a *hierarchically distributed* set of decision makers. This capability will allow the representation of various levels of battlefield decision making (strategic, operational, and tactical) in a more realistic manner. Such a capability would also provide an excellent test-bed for evaluating distributed cooperative expert systems.

Several interface modules were built using the core set of tools provided by LOTTA. KEELA linked LOTTA to the KEE expert system tool by converting Flavor instances to KEE units while KEE provided escapes to Lisp that could call the LOTTA tools directly. The current system, LISA, is both more simple and more efficient as it links LOTTA directly to the Tactical Aerial Situation Assessment System, described in Section 3.5.2.

3.4 Reasoning System

The simulation environment generates enough complex situations to exercise the requirements of several crucial tasks in battlefield management: information fusion, situation assessment, option generation and assessment, and decision evaluation and execution. The reasoning capabilities are not embedded in the simulation, but are instead part of a separate expert system that will aid or take the role of one the players. This architecture allows us to generate a scenario, analyze and assess it using inference techniques, make a decision based on the situation assessment, execute the decision by issuing the proper commands (messages) to the simulated objects, change the scenario, and continue the loop.

In part, the separation of the reasoning system from the simulation environment has been due to the need of addressing the increased complexity induced by the presence of uncertainty in the dynamic classification problem. Uncertainty can be generated by the sources of information, information is always of limited reliability: images may be blurry or partially occluded, text messages are ambiguous, and the information may be intentionally misleading (i.e., projection of false images). The problem solving knowledge used in these domains is itself intrinsically uncertain: the interpretation of the numerous patterns is based on subjective predicates, and the conclusions derived from the recognition of given patterns are plausible but not categorical. The other reason for separating the reasoning system from the simulation environment is the software engineering problem associated with deriving a knowledge base for the reasoning system.

3.4.1 AI Software Engineering Problem

Usually, dynamic classification problems are characterized by an evolving set of requirements. As a result, their developments undergo a large number of iterative refinements, as described by the rapid prototyping paradigm [Pre87]. The prototypes are developed in rich and flexible environments in which various AI techniques are used. A knowledge base is generated, debugged, modified, and tested until a "satisficing" solution [Sim81] is obtained from this development phase. Then the prototype is ready for deployment: it is ported to specific platforms and embedded into larger systems. The deployment's success, however, depends on the application performing in *real-time*. If the reasoning system does not provide good timely information, then the application will not be able to react fast enough to its environment. Even after deployment, the prototype cycle must continue, because performance verification can only take place in a real-time en-

vironment. Thus, in order to meet the real-time requirements, the knowledge base and algorithms may need additional prototyping.

AI software development is significantly different from the traditional approach. It requires a prototyping cycle which spans two environments: development and target. Usually, instead of having to transition software between these two environments, one environment is eliminated. This approach, however, compromises either the flexibility and richness needed for development, or the speed and efficiency requirements of execution. When both environments are used, a smooth transition of the application between these two environments is essential. If the prototyping cycle cannot completely span the two environments, the knowledge engineer has to re-implement portions of the software.

The reasoning tool described in this section provides a rich, user-friendly development environment, a small and quick run-time system, and translation software to span the two (see Figure 3.4).

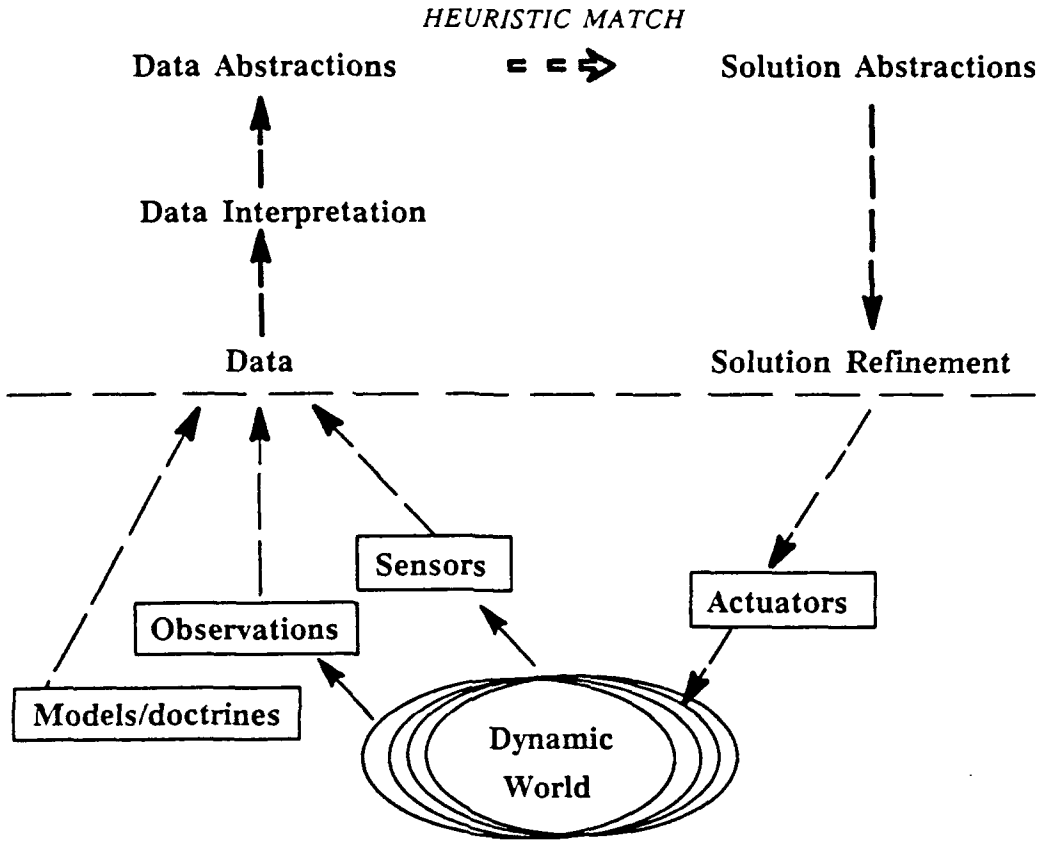


Figure 3.1: The Dynamic Classification Problem

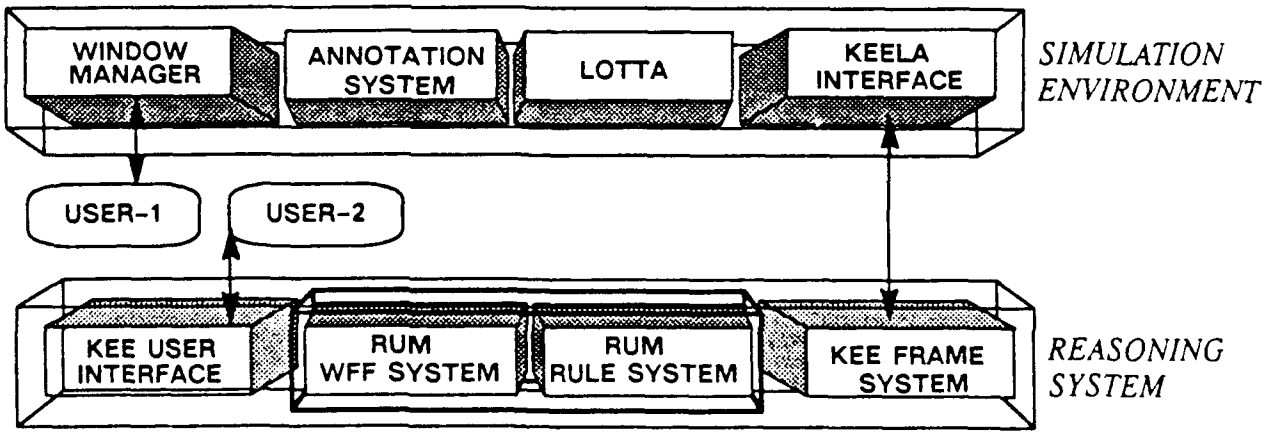


Figure 3.2: The Test-bed Architecture

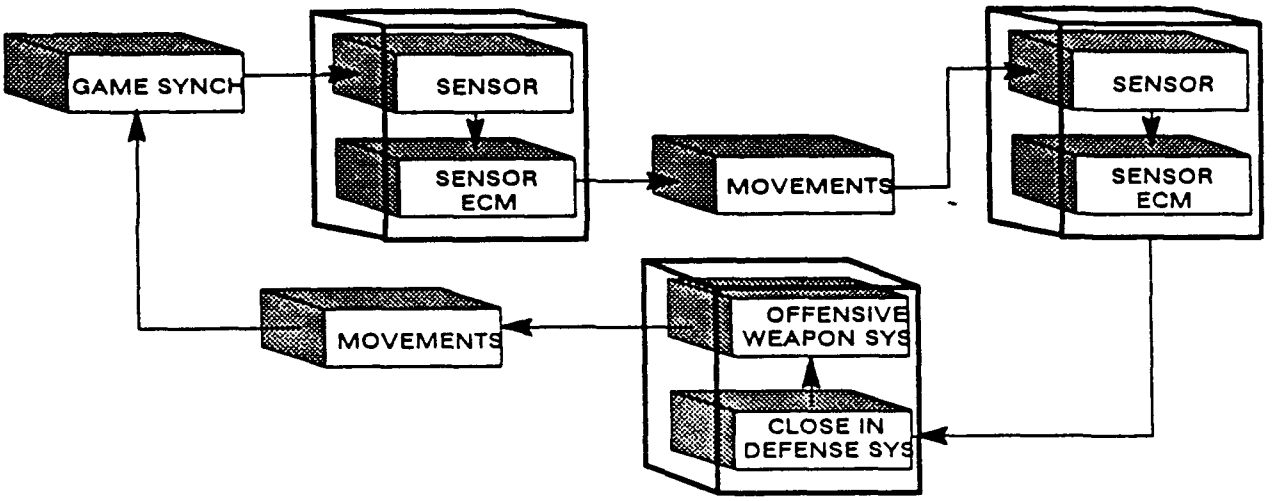


Figure 3.3: Simulation Cycle in LOTTA

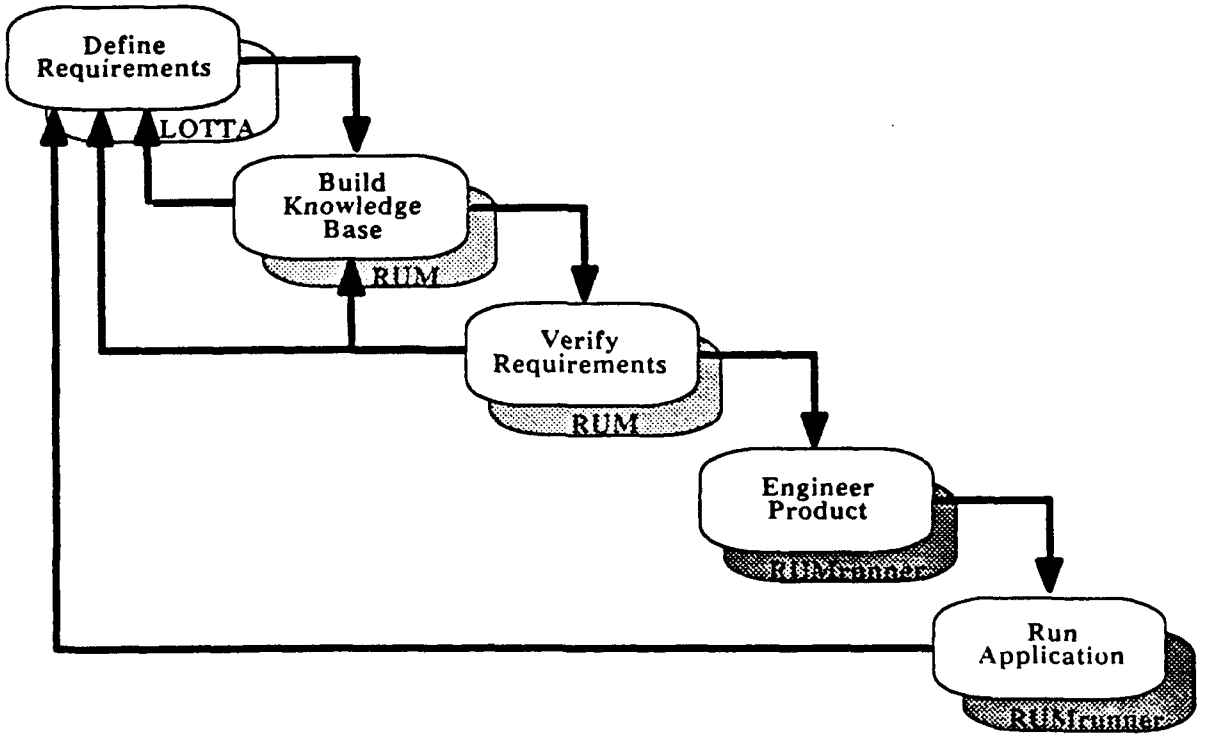


Figure 3.4: Software Engineering with RUM and RUMrunner

The Reasoning with Uncertainty Module (RUM) [5] [RUM87] allows the knowledge engineer to build complex applications in a user-friendly, error-tolerant, mouse-and-menu environment. This environment also makes available many artificial intelligence techniques, including reasoning with uncertainty. The RUMrunner tool provides a small, fast, streamlined run-time system along with a virtually transparent transition path from the development environment. These two tools allow the knowledge engineer to build the prototype and deploy the final application in the most desirable environments.

3.4.2 RUM

RUM, a development environment for reasoning with uncertainty, and RUMrunner, RUM's run-time counterpart, are the reasoning systems used in the test-bed architecture. RUM [5] is based on Bonissone's theory of plausible reasoning [2], which provides a representation of uncertain information, uncertainty calculi for inferencing, and selection of calculi for inference control. Uncertainty is represented in both facts and rules. A fact represents the assignment of a value to a variable. A rule represents the deduction of a new fact (conclusion) from a set of given facts (premises). Facts are qualified by a degree of *confirmation* and a degree of *refutation*. Rules are discounted by *sufficiency*, indicating the strength with which the premise implies the conclusion, and *necessity*, indicating the degree to which a failed premise implies a negated conclusion. The uncertainty present in this deductive process leads to considering several possible values for the same variable. Each value assignment is qualified by different uncertainties, which are combined with special calculi as described in [3] and [4].

RUM's rule-based system integrates both procedural and declarative knowledge in its representation. The rule-based approach captures expertise gained from experience or "rules of thumb", thereby codifying heuristic knowledge without any underlying model. In addition, natural expression of procedural knowledge can be smoothly integrated through user-defined predicates in RUM rules. The integration of both techniques is essential to solve situation assessment problems, which involve both heuristic and procedural knowledge.

The expressiveness of RUM is further enhanced by two other functionalities: the *context mechanism* and *belief revision*. The context represents the set of preconditions determining the rule's applicability to a given situation. This mechanism provides an efficient screening of the knowledge base by focusing the inference process on small rule subsets. For instance, in SA, selected rules describe the behavior of friendly planes, while others should only be applied to unfriendly or unidentified ones. The rule's context provides this filtering mechanism.

RUM's belief revision is essential to the dynamic aspect of the classification problem. The belief revision mechanism detects changes in the input, keeps track of the dependency of intermediate and final conclusions on these inputs, and maintains the validity of these inferences. For any conclusion made by a rule, the mechanism monitors the changes in the certainty measures that constitute the conclusion's support. Validity flags are used to reflect the state of the certainty. For example, a flag can indicate that the uncertainty

measure is valid, unreliable (because of a change in the support), too ignorant to be useful, or inconsistent with respect to the other evidence.

RUM offers both backward and forward processing. A *lazy evaluation*, running in backward mode, recomputes the certainty measures of the minimal set of facts required to answer a given query. This mode is used when the system or the user decide that they are dealing with time-critical tasks. Breadth-first, forward mode processing recomputes the certainty measures attempting to restore the integrity of the rule deduction graph. This mode is used by the system when time is not critical.

These AI capabilities are used to develop a knowledge base, in conjunction with RUM's software engineering facilities, such as flexible editing, error checking, and debugging. Some of these features, however, are no longer necessary once the development cycle is complete. At run-time, applications do not create new knowledge (facts or rules), as their basic structure have been determined at compile-time. The only run-time requirement is the ability to instantiate rules and facts from their pre-determined definitions. By eliminating the development features which are unnecessary at run-time, a real-time AI system can improve upon the algorithms and methodologies used in RUM.

3.4.3 RUMrunner

The objective of RUMrunner [Pfa87] is to provide a software tool that transforms the customized knowledge base generated by the development phase, into a fast and efficient real-time application. RUMrunner provides both the functionality to reason about a broad set of problems, and the speed required to properly use the results of the reasoning process. Performance improvements are obtained by implementing all RUM's functionalities with leaner data structures, using Flavors [Sym86] (for the Symbolics version) or *defstructs* (for the Sun version). Furthermore, RUMrunner no longer requires the use of the KEE software, thus it can be run on any Symbolics or Sun workstation with much smaller memory configurations, and without a KEE software license. RUMrunner's inference engine also provides a scheduling mechanism, a planning algorithm for reasoning under time pressure, and other functionalities needed by real-time applications. RUMrunner has four major qualities: it provides a meaningful subset of AI techniques, it runs fast, it has the functionality of a real-time system, and it does not require the software engineer to re-program the application in the target environment.

To increase speed, RUMrunner takes advantage of the fact that the application has been completely developed and debugged. It provides a minimum of error checking because the application is assumed either to be debugged already, or to be robust enough to handle errors. RUMrunner's time performance in reasoning tasks is partially due to the compilation of the knowledge base. As a result of this compilation, new or different rules or units cannot be created in the knowledge base after the translation.

RUMrunner provides additional functionality for applications which must satisfy real-time requirements. A RUMrunner application is able to carry out and control a set of activities to rapidly respond to its environment. To meet these goals, the interface of RUMrunner with the application program is designed to be asynchronous, allowing the

application to avoid unnecessary delays. In addition, the application is able to handle externally or internally driven interrupts. It is also able to prioritize tasks, by using an agenda mechanism [Erm80], so that RUMrunner handles the most important ones first. RUMrunner is performance-conscious by ensuring that tasks execute within a specified amount of time. This is done through planning the execution of a single task as suggested by Durfee and Lesser [DL87]. Finally, RUMrunner is implemented in Common LISP, thus it can be ported to many machines without requiring any proprietary software. None of this additional functionality takes an unreasonable amount of time, and if not desired, most of it can remain unused without a great time penalty. RUMrunner, is further elaborated upon in [Pfa87].

3.5 Using the Test-bed Architecture

In section 2.4 we described two experiments used to exercise the test-bed architecture. For the reader's convenience, we provide a summary of them again.

3.5.1 Information Fusion and Platform Typing in a Naval Scenario

The first experiment dealt with a naval scenario and has been reported in [4]. The experiment was a modified version of the naval situation assessment scenario used by Naval Ocean System Command to test STAMMER [BM79] and STAMMER2 [MMK79]. In this modified scenario, a CGN-36 missile cruiser operating a passive sensor and an SPS-10 surface radar faces two unknown platforms. One of the two platforms (selected from a large set of ships) is using an active sensor (navigational radar), while the second platform is not using any active sensor.

The cruiser's task was to track, correlate, and classify each detected object. The passive and active sensors were turned on, generating sensor reports which were translated through the KEELA interface into observed *wffs*. The information returned by the passive sensor contained the heading, position, range, speed, and time at which the platform was detected. This information was attached to a track which maintained subsequent sensor reports generated by the same sensor and associated with the same platform. A second track for the platform was similarly generated by the SPS-10 radar. A third track, also generated by the cruiser's active sensor, was generated for the second platform.

The query posed to RUM was to deduce the class of the first platform using the sensor tracks. Using the RUM knowledge base and backward chainer, various attributes of the platform were inferred or observed. The platform was correctly identified as a merchant ship, based on the fact that the platform was: reasonably close to a shipping lane; traveling at a typical freighter speed (in the 9-14 miles/hour range); not maneuvering; and not trying to dodge the cruiser's surface radar. Three values for the platform classes were considered by the system and qualified by their corresponding certainty bounds: *Merchant* [0.69 1], *Submarine* [0 0.2], and *Fishing Boat* [0 0.02], *Merchant* being best because of the ranking of certainty measures. The lower bound of 0.69 indicates a

large amount of positive (confirming) evidence. The upper bound of 1.0 indicates the absence of any negative (refuting) evidence. The class *Submarine* obtained no confirming evidence and a large amount of negative evidence. The refuting evidence was provided by a rule which from the failure to observe a close-distance radar pop-up determined that there was only a small chance for the platform to be a submarine. The class *Fishing Boat* also had no confirming evidence and an overwhelming amount of negative evidence. This refuting evidence was due to the fact that the platform was too far from the fishing areas, too big for a fishing boat, and was using a radar.

3.5.2 Tactical Aerial Situation Assessment

The second experiment dealt with tactical aerial situation assessment. The purpose of the experiment was to provide a fighter pilot with the intent evaluation of various potential threats. The simulator generated a variety of scenarios in which up to three aircraft exhibited sufficiently interesting behavior (flight paths intercepting/converging toward ownship, specific sensor use, etc.) to justify a closer analysis. RUM deduced the aircraft's intent from a variety of factors. First the aircraft's class and type was identified by a set of rules based on behavioral information. This inference determined characteristics such as a likely weapon configuration, a likely sensor configuration and an estimate of the Launch Acceptability Region (LAR). Intent was then determined by a second set of RUM rules, based on aspect angle, change in aspect angle, velocity, acceleration, radar mode, ownship detectability template (ODT), shortest time to threat's LAR, and formation. In this experiment, the reasoning system correctly evaluated various intent values chosen among *engage-now*, *engage-later*, *influence*, *evade*, and *non-reactive*. Each plausible intent value was qualified by an uncertainty measure and, from the induced partial ordering, the most likely intent was returned.

3.6 Testing and Validating

Figure 3.4, in Section 3.4.1, illustrates the cascading tasks associated with the development of a knowledge base application. The first three tasks (*Requirement Re-definition*, *KB Development*, *Requirement Verification*) are performed in the development environment. The last two tasks (*Product Engineering and Performance Verification*) are performed in the deployment system.

3.6.1 Functional Validation

The objective of this task is to assure that the knowledge base will meet the requirements derived from the problem definition. We have used LOTTA to generate a set of scenarios (sequence of events), which collectively exercise all the desired requirements. For instance, these scenarios have allowed us to test the KB in light of unexpected events, such as the appearance of a second platform in a one-on-one situation, or while reasoning with

reduced information to reflect constraints on the use of the own-ship's active sensors, etc. By interactively modifying LOTTA's scenarios, we have tested the reasoning system on a class of scenarios with multiple variations, representing "what-if" type of situations.

In all these scenarios, LOTTA maintained ground truth (i.e., states and sets of orders of all the players' objects.) At the end of each sensor phase, LOTTA generated the corresponding track file information representing the perceived truth of the simulated world. These track files have then been used to test the rule set for consistency and completeness. The same track files, stored as buffers, have later been applied as probing input to exercise the run-time system.

RUM's conclusion's explanation and traceability facilities have been used to identify and analyze the dominant rules responsible for specific conclusions. By comparing the conclusions with ground truth, the knowledge engineer has been able to detect and correct eventual discrepancies. This corrective process was achieved by verifying the validity of the input to the rule set (track file information), by examining the context of the active rules, by analyzing the structure of the active rules (under or over constrained), by calibrating the strength of the dominant rules (sufficiency and necessity), and by modifying the sensitivity to uncertainty exhibited by the dominant rules (uncertainty calculus selection).

3.6.2 Performance Validation

The objective of this task is to guarantee that the software will meet the timing requirements imposed by the real-time constraints, while still maintaining the same functional behavior.

As described in Section 3.4.3, this goal was achieved by a combination of efforts: the translation of RUM's complex data structure into simpler, more efficient ones (to reduce overhead); the compilation of the rule set into a modified RETE net [For82] [Mir87] (to avoid run-time search); the load-time estimation of each rule's execution cost (to determine, at run-time, the execution cost of any given deductive path); the run-time planning mechanism for model selection (to determine the largest relevant rule subset which could be executed within a given time-budget).

3.6.3 Example of Testing and Validating a KB

Using the Tactical Aerial Situation Assessment scenario discussed in Section 3.5.2, we will illustrate how the test-bed architecture has been applied to this problem.

The original Tactical Aerial Situation Assessment Module was built with RUM on a Symbolics running the KEE software. First the RUM system is loaded, and then the KEE knowledge base is created. Units such as `plane` are created, designed to be instantiated at run-time for each observed plane. RUMrules are created to infer the target value, threat value, radar range of objects, and to identify the primary and secondary mission targets. These rules either describe attributes of single planes or define relationships among various planes (e.g., formation). These rules are designed to be instantiated

at run-time along with the units, when particular planes are detected by the available sensors. After the units and rules are created, the knowledge base is debugged and fine-tuned by modifying the certainties of values and rules, as well as the structure of the rules. Scenarios are generated using LOTTA to provide realistic input data to the system. As new requirements are added, new rules are created. Finally, after further testing and debugging, the system is verified by the pilot experts.

At this point, when development is finally complete, the application is ready for RUMrunner. The goal is now to ensure that the system meets the real-time requirements.

Using RUMrunner, the knowledge base is automatically translated into a binary file. This point marks the end of the dependency on the KEE system. The RUMrunner system, the application software, and the RUMrunner application knowledge base are loaded into a (potentially) different Symbolics machine or Sun workstation. This process is illustrated in Figure 3.5.

After testing the application (with the data generated from the LOTTA simulations) to ensure its correct behavior, the real-time functionalities are added to the system. A second real-time binary file is created after RUMrunner manipulates the application knowledge base to extract the real-time information. Finally, after loading the second file, the system can be in run-time mode.

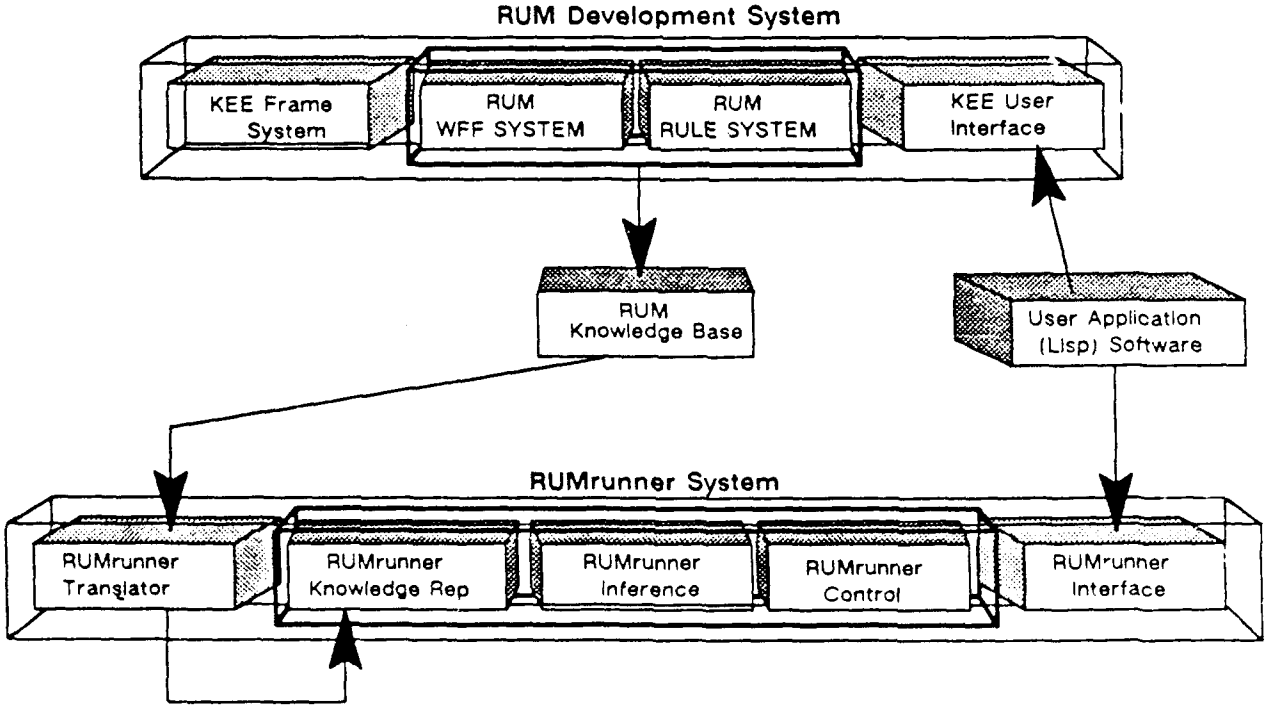


Figure 3.5: Transitioning from Development to Deployment

The only alteration to the application software is made by calling a single function, which identified the RUMrunner tasks which were time-critical and augmenting the corresponding queries with the appropriate time budget. The application is run, and its time performance measured, resulting in some of the application functions running more than 200 times faster than those in the original RUM application. If the system is not meeting its real-time requirements, the bottlenecks are identified and the system is fine-tuned.

3.6.4 Software Portability

Currently, RUM runs on top of KEE on Symbolics and Sun Workstations. RUMrunner runs on Symbolics and SUN workstations with Lucid Common LISP. We are now exploring the porting of RUMrunner to Microvax and Masscomp workstations. We are also developing an Ada version of RUMrunner, running on the Sun workstations, which will be rule compatible with its Common LISP version.

3.7 Conclusions

In this paper, we have described the implementation of a simulation environment centered around LOTTA, a symbolic simulator written in Flavors, and a reasoning system, RUM, capable of reasoning with uncertain information.

LOTTA provides the environment for simulating time-varying scenarios. RUM allows the application to be built in a rich development environment, and then, using its run-time counterpart RUMrunner, cross-compiles the knowledge into a more efficient form. The compiled knowledge runs on an efficient driver so that modifications to the application software are not required. Through planning on the compiled reasoning graph of facts and rules, RUMrunner ensures that reasoning can be performed in the application within an allotted amount of time. In addition, the resulting application can be asynchronous and interruptible, to allow the system to be embedded into a larger real-time application.

The combination of LOTTA and RUM has proven adept at verifying the rule set and functionality of applications, which require reasoning in complex, changing environments.

Bibliography

[BB85] Piero P. Bonissone and Allen L. Brown. Expanding the Horizons of Expert Systems. In Thomas Bernold, editor, *Expert Systems and Knowledge Engineering*, pages 267–288, North-Holland, Amsterdam, 1986.

[BGD87] Piero P. Bonissone, Stephen Gans, and Keith S. Decker. RUM: A Layered Architecture for Reasoning with Uncertainty. In *Proceedings 10th International Joint Conference on Artificial Intelligence*, pages 891–898, AAAI, August 1987.

[BM79] R. J. Bechtel and P. H. Morris. *STAMMER: System for Tactical Assessment of Multisource Messages, Even Radar.* Technical Report 252 (NTIS ADA116527), Naval Ocean System Command, S.Diego, CA, May 1979.

[Bon87] Piero P. Bonissone. Plausible Reasoning: Coping with Uncertainty in Expert Systems. In Stuart Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 854–863, John Wiley and Sons Co., New York, 1987.

[Bon87b] Piero P. Bonissone. Summarizing and Propagating Uncertain Information with Triangular Norms. *International Journal of Approximate Reasoning*, 1(1):71–101, January 1987.

[Bon87c] Piero P. Bonissone. Using T-norm Based Uncertainty Calculi in a Naval Situation Assessment Application. In *Proceedings of the the Third AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 250–261, AAAI, July 1987.

[Cla84] William J. Clancey. Classification Problem Solving. In *Proceedings Third National Conference on Artificial Intelligence*, AAAI, August 1984.

[DL87] Edmund H. Durfee and Victor R. Lesser. *Planning to meet deadlines in a blackboard-based problem solver.* Technical Report COINS-87-07, COINS at University of Massachusetts, 1987.

[Erm80] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D.R. Reddy. The Hearsay-II Speach-Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12:213–253, 1980.

[FH87] R. J. Firby and S. Hanks. A Simulator for Mobile Robot Planning. In *Proceedings DARPA Knowledge-Based Planning Workshop*, pages (23–1)—(23–7), DARPA, December 1987.

[For82] Charles L. Forgy. Rete: A fast Algorithm for the many pattern/many object pattern match problem. *Journal of Artificial Intelligence*, 19(1):17–37, September 1982.

[KEGN86] P. Klahr, J. W. Ellis, W. D. Giarla, S. Narain andE. M. Cesar, and S. R. Turner. TWIRL: Tactical Warfare in the ROSS Language. In P. Klahr and D. Waterman, editors, *Expert Systems: Techniques, Tools, and Applications*, pages 224–268, Addison Wesley, Reading, Ma, 1986.

[KMN82] P. Klahr, D. J. McArthur, and S. Narain. SWIRL: An Object-Oriented Air Battle Simulator. In *Proceedings Second National Conference on Artificial Intelligence*, pages 331–334, AAAI, August 1982.

[Mir87] Daniel P. Miranker. TREAT: A better match algorithm for AI production systems. In *Proceedings Sixth National Conference on Artificial Intelligence*, pages 42–47, AAAI, July 1987.

[MKN86] D. J. McArthur, P. Klahr, and S. Narain. ROSS: An Object-oriented Language for Constructing Simulations. In P. Klahr and D. Waterman, editors, *Expert Systems: Techniques, Tools, and Applications*, pages 70–91, Addison Wesley, Reading, Ma, 1986.

[MMK79] D. C. McCall, P. H. Morris, D.F. Kilber, and R. J. Bechtel. *STAMMER2 Production System for Tactical Situation Assessment, Vol 1,2*. Technical Report 298 (NTIS ADA084038 ADA084053), Naval Ocean System Command, S.Diego, CA, October 1979.

[Pfa87] Lise M. Pfau. *RUMrunner: Real-Time Reasoning with Uncertainty*. Master's thesis, Rensselaer Polytechnic Institute, December 1987.

[Pre87] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, second edition, 1987.

[RUM87] *RUM User Guide*. General Electric Company, 1987. Version 11.0.

[SBBG86] L. M. Sweet, P. P. Bonissone, A. L. Brown, and S. Gans. Reasoning with Incomplete and Uncertain Information for Improved Situation Assessment in Pilot's Associate. In *Proceedings of the 12th DARPA Strategic Systems Symposium*, DARPA, October 1986.

[Sim81] Herbert A. Simon. *The Sciences of the Artificial*. M.I.T. Press, Cambridge, Massachusetts, second edition, 1981.

[Sym86] *Symbolics Common Lisp - Language Concept (Symbolics Documentation Set, vol. 2A).* Symbolics, 1986. Genera 7.0.

4. New Results on Semantical Nonmonotonic Reasoning

Allen L. Brown, Jr.
GE Corporate Research and Development Center
P.O. Box 8
Schenectady, New York 12301

Yoav Shoham
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

In earlier reports we presented a semantical account of nonmonotonic reasoning based on the partial ordering of interpretations of standard logics. In this article we generalize and extend the earlier work. We elucidate the structural relation between the new work and the old. Finally, we apply the new results to give a logical semantical account of justification-based truth maintenance.

4.1 Introduction

In [Sho86] a general semantical framework for constructing nonmonotonic logics was developed. While this framework, based purely on partial orders on models standard logics, does not capture all nonmonotonic logics, it does elucidate many of the better known such logics, and serves as a basis for capturing the others. In this paper we augment and generalize the previous work in three ways.

1. We investigate an alternative formulation, in which the relation on models is reflexive and transitive, but not necessarily a partial order (see below).

2. Relative to the first augmentation, we show a natural way in which to define stratified nonmonotonic logics within the semantical framework.

3. We show how existing "truth maintenance" systems can be given a precise account within our enlarged framework.

The following three sections deal with each of these three issues respectively. In the remainder of this section we review the construction offered in [Sho86], so as to make this article self contained.

As defined in [Sho86], a *preferential logic* is the logic $\mathcal{L}_{\sqsubset}$, where $\mathcal{L}$ is any standard (propositional or first order, classical or modal) logic, and $\sqsubset$ is any partial order on the interpretations (or models) of $\mathcal{L}$. Intuitively, $\sqsubset$ can be thought of as the "preferred model" relation, so that the intuitive reading of $M \sqsubset M'$ is "M' is preferred over M." Several

formal properties were defined for preferential logics, such as satisfaction, satisfiability, validity and entailment. Here we shall reconstruct only some of them.

Definition 4.1.1 *A model M* preferentially satisfies (p-satisfies) *a sentence A in $\mathcal{L}_\sqsubset$ (written $M \models_\sqsubset A$) if and only if $M \models A$, and there is no other model M' such that $M \sqsubset M'$ and $M' \models A$* [1]

Definition 4.1.2 *Let A and B be two sentences in $\mathcal{L}_\sqsubset$. A is said to* preferentially entail (p-entail) *B (written $A \models_\sqsubset B$) if and only if for any M, if $M \models_\sqsubset A$ then $M \models B$. In other words, $A \models_\sqsubset B$ if and only if B is true in all preferred models of A.*

Definition 4.1.3 *$\mathcal{L}_\sqsubset$ is* preferentially monotonic (p-monotonic) *if and only if for any A, B $C \in \mathcal{L}_\sqsubset$, if $A \models_\sqsubset C$ then $A \wedge B \models_\sqsubset C$.*

Some preferential logics are monotonic (such as when one selects the empty partial order), and many are not (such as those resulting from selecting the partial order implicit in a circumscription axiom). We end this section with the following characterization of p-monotonicity:

Definition 4.1.4 *A partial order $\sqsubset$ is* complete *if and only if for every (possibly infinite) sequence of models $M_1 \sqsubset M_2 \sqsubset \cdots \sqsubset M_i \sqsubset \cdots$ there exists a model M that is an upper bound for the sequence (that is, $M_i \sqsubset M$ for every i in the sequence such that $M_i \neq M$) and there is no upper bound M' for the the sequence such that $M' \sqsubset M$.*

Proposition 4.1.1 *For any preferential logic $\mathcal{L}_\sqsubset$ such that $\sqsubset$ is complete, $\mathcal{L}_\sqsubset$ is monotonic if and only if $\sqsubset$ is the empty relation.*

4.2 Biased Logics

The reader may have noticed that p-satisfiability, p-entailment and p-monotonicity would be well defined even if $\sqsubset$ were not a partial order, although then we would lose the intuitive meaning of that relation. In this section we investigate a slightly different restriction on the binary relation that still makes intuitive sense.

Specifically, we propose replacing the $\sqsubset$ by any binary relation $\sqsubseteq$ that is reflexive and transitive. We also replace the intuitive reading of $M \sqsubset M'$ as "M' is better than M" by the intuitive reading of $M \sqsubseteq M'$ as "M' is at least as good as M." We then relate the new construction to the previous one, the intuition being that "M' is better than M" just in case "M' is at least as good as M, but M is *not* at least as good as M'."

As in the definitions in preferential logics, when in the following we speak of a "standard logic" we mean any of the customary monotonic logics (*e.g.*, propositional or first order, classical or modal, where in the modal case we allow any structure of possible worlds). When we speak of an interpretation or a model in a standard logic, we mean that which goes to the left of the $\models$ relation in that logic.

[1] In [Sho86] p-satisfiability was called simply *satisfiability*, but here we shall want to distinguish it from the related notion of *b-satisfiability*. The same applies to *p-entailment* and *p-monotonicity* below.

Definition 4.2.1 *A* biased logic *is a logic $\mathcal{L}_{\sqsubseteq}$ where $\mathcal{L}$ is any standard logic, and $\sqsubseteq$ is a reflexive and transitive binary relation on interpretations of $\mathcal{L}$.*

The syntax of $\mathcal{L}_{\sqsubseteq}$ is identical to the syntax of $\mathcal{L}$. Next we define the semantics.

Definition 4.2.2 *Let M be an interpretation in $\mathcal{L}$ and A a sentence. M* biasedly satisfies *(*b-satisfies*) A (written $M \models_{\sqsubseteq} A$) if and only if*

1. *$M \models A$, and*
2. *there is no M' such that*
   (a) *$M \sqsubseteq M'$,*
   (b) *$M' \not\sqsubseteq M$, and*
   (c) *$M' \models A$.*

Definition 4.2.3 *Let A, B be sentences in $\mathcal{L}_{\sqsubseteq}$. A* biasedly entails *(*b-entails*) B (written $A \models_{\sqsubseteq} B$) if and only if for any M, if $M \models_{\sqsubseteq} A$ then $M \models B$.*

Definition 4.2.4 *$\mathcal{L}_{\sqsubseteq}$ is* biasedly monotonic *(*b-monotonic*) if and only if for any $A, B, C \in \mathcal{L}_{\sqsubseteq}$, if $A \models_{\sqsubseteq} C$ then $A \wedge B \models_{\sqsubseteq} C$.*

Analogous to our earlier characterization of p-monotonicity b-monotonicity is characterized as follows:

Definition 4.2.5 *A reflexive, transitive binary relation $\sqsubseteq$ is* complete *if and only if for every (possibly infinite) sequence of models $M_1 \sqsubseteq M_2 \sqsubseteq \cdots \sqsubseteq M_i \sqsubseteq \cdots$ there exists a model M that is an upper bound for the sequence (that is, $M_i \sqsubseteq M$ for every i in the sequence) and there is no upper bound M' for the the sequence such that $M' \sqsubseteq M$ and $M \not\sqsubseteq M'$.*

Proposition 4.2.1 *For any biased logic $\mathcal{L}_{\sqsubseteq}$ such that $\sqsubseteq$ is complete, $\mathcal{L}_{\sqsubseteq}$ is monotonic if and only if $\sqsubseteq$ is the equivalence relation.*

Biased logics are closely related to preferential logics. In fact, we show that one can translate freely between these two types of logics, preserving the notions of entailment and monotonicity.

4.2.1 From preferential logic to biased logic

The first translation is trivial. Given a preferential logic $\mathcal{L}_{\sqsubset}$ we construct the biased logic $\mathcal{L}_{\sqsubseteq}$, where $\sqsubseteq$ is the reflexive closure of $\sqsubset$. Observe that $\sqsubseteq$ is an equivalence relation only if $\sqsubset$ is empty, in which case $\sqsubseteq$ would be the identity relation.

Proposition 4.2.2 *For any A, B, $A \models_{\sqsubset} B$ in $\mathcal{L}_{\sqsubset}$ if and only if $A \models_{\sqsubseteq} B$ in $\mathcal{L}_{\sqsubseteq}$.*

Corollary 4.2.1 *$\mathcal{L}_{\sqsubset}$ is p-monotonic if and only if $\mathcal{L}_{\sqsubseteq}$ is b-monotonic.*

4.2.2 From biased logic to preferential logic

This second translation is only slightly more elaborate. It is helpful here to have in mind the graph-theoretic interpretation of biased logics. Each such logic $\mathcal{L}_{\sqsubseteq}$ defines a directed graph $G(V, E)$, where V is the set of all interpretations of $\mathcal{L}$, and $E = \{\langle M_1, M_2\rangle | M_1 \sqsubseteq M_2\}$. We can identify the *strongly connected components* [AHU74] of G, each being a set of vertices any two of which are connected via a directed path (in both directions). In our case, since $\sqsubseteq$ is transitive, we have that any two vertices in a strongly connected component are in fact directly connected by an edge. In other words, each strongly connected component is a *complete* directed graph. For a similar reason, we have that if M_1 is a vertex in a component C_1, M_2 is in C_2, and $\langle M_1, M_2\rangle \in E$, then for *any* $M_1' \in C_1, M_2' \in C_2$, $\langle M_1', M_2'\rangle \in E$. Now consider the so-called *super graph* of G, $G'(V', E')$. V' consists of the strongly connected components of G, and $\langle C_1, C_2\rangle \in E'$ if and only if there are directed edges in G connecting the vertices in C_1 to the vertices in C_2. It is not hard to see that G' must be acyclic, or, in other words, that E' is a strict partial order. With this intuition, and given a biased logic $\mathcal{L}_{\sqsubseteq}$, we construct a preferential logic as follows:

Definition 4.2.6

a. $M \sqsubset M'$ *if and only if* $M \sqsubseteq M'$ *and* $M' \not\sqsubseteq M$.

b. $M \sim M'$ *if and only if* $M \sqsubseteq M'$ *and* $M' \sqsubseteq M$.

Lemma 4.2.1 $\sqsubset$ *is a strict partial order, and* $\sim$ *is an equivalence relation.*

Proposition 4.2.3 *For any* A, B, $A \models_{\sqsubseteq} B$ *in the biased logic* $\mathcal{L}_{\sqsubseteq}$ *if and only if* $A \models_{\sqsubset} B$ *in the preferential logic* $\mathcal{L}_{\sqsubset}$.

Corollary 4.2.2 $\mathcal{L}_{\sqsubseteq}$ *is b-monotonic if and only if* $\mathcal{L}_{\sqsubset}$ *is p-monotonic.*

4.3 Stratifying nonmonotonic logics

In the previous section we showed how an apparent change in the logic in fact leaves its expressiveness unchanged, although for some applications the new form will be more convenient. Here we discuss another such augmentation, that is a very convenient one, but which again does not complicate the properties of the logic.

In the construction so far, whether in the original formulation of preferential logics or the new one of biased logics, we started by saying "start with a standard logic ...." We now propose to start with *any* logic, possibly a nonmonotonic one, and thus "stack" nonmonotonic logics one on top another. This will be convenient for many purposes. One example arises when we formalize truth maintenance systems. Another is logic programming, where in determining the semantics of the negation operator, it is computationally important whether or not the programs are "stratified." Roughly speaking

a logic program is stratified if it can be decomposed into layers such that one layer refers only to predicates appearing in lower layers. Although we shall not discuss logic programming any further here, we shall use the term *stratified logics* below, reflecting the strong connection between our construction and these issues that are being actively investigated in the logic programming community [Min87].

Definition 4.3.1 *Given a standard logic $\mathcal{L}$ and a set $B = \{\sqsubseteq_i\}$ of reflexive and transitive binary relations on interpretations of $\mathcal{L}$, the set of* stratified logics *is defined inductively as follows.*

1. *$\mathcal{L}_{\sqsubseteq}$ is a stratified logic.*
2. *If $\mathcal{L}_X$ is a stratified logic and $\sqsubseteq_i \in B$, then $\mathcal{L}_{(X\sqsubseteq_i)}$ is a stratified logic. Since the notation is unambiguous, we shall sometimes drop the parentheses. For example, might substitute $\sqsubseteq_1\sqsubseteq_{i2}\sqsubseteq_{i3}$ for $((\sqsubseteq_{i1}\sqsubseteq_{i2})\sqsubseteq_{i3})$.*
3. *There are no other stratified logics.*

The syntax of all these stratified logics is identical to that of $\mathcal{L}$. Their semantics are defined as follows. For every stratified logic $\mathcal{L}_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}}$ we define a relation $\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}$ on interpretations of $\mathcal{L}$. This relation can be viewed as the iterative refinement of the individual relations. Specifically, we make the following inductive definition:

Definition 4.3.2 *Let $\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}$ be as above, and M_1, M_2 two interpretations. $M_1 \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n} M_2$ if and only if one of two conditions holds:*

1. *$M_1 \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_{n-1}} M_2$ but it is not the case that $M_2 \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_{n-1}} M_1$.*
2. *$M_1 \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_{n-1}} M_2$, $M_2 \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_{n-1}} M_1$, and $M_1 \sqsubseteq_{i_n} M_2$.*

Definition 4.3.3 *Let $\mathcal{L}_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}}$ be a stratified logic. An interpretation M* stratifiedly satisfies (s-satisfies) *A in $\mathcal{L}$ (written $M \models_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}} A$) if and only if*

1. *$M \models A$, and*
2. *there is no other M' such that*
   (a) *$M \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n} M'$,*
   (b) *it is not the case that $M' \sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n} M$, and*
   (c) *$M' \models A$.*

Definition 4.3.4 *Let $\mathcal{L}_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}}$ be a stratified logic, and A, B two sentences in it. A* stratifiedly entails *(s-entails) B in $\mathcal{L}$ (written $A \models_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}} B$) if and only if for any interpretation M, if $M \models_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}} A$ then $M \models B$.*

Definition 4.3.5 $\mathcal{L}_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}}$ *is* stratifiedly monotonic (s-monotonic) *if and only if for any* A, B, C, *if* $A \models_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}} C$ *then* $A \wedge B \models_{\sqsubseteq_{i_1}\cdots\sqsubseteq_{i_n}} C$.

Finally, we note that stratified logics are not a radical departure from biased ones. In fact, every biased logic is stratified and *vice versa.*

Lemma 4.3.1 *For any* $\sqsubseteq_{i_1}, \cdots, \sqsubseteq_{i_n}$ *as above,* $\sqsubseteq_{i_1} \cdots \sqsubseteq_{i_n}$ *is both reflexive and transitive.*

Lemma 4.3.2 $\sqsubseteq_{i_1} \cdots \sqsubseteq_{i_n}$ *is complete if and only if each of the* $\sqsubseteq_{i_j}$ *is complete.*

Corollary 4.3.1 *Let* $\sqsubseteq_{i_1} \cdots \sqsubseteq_{i_n}$ *be complete. Then* $\sqsubseteq_{i_1} \cdots \sqsubseteq_{i_n}$ *is b-monotonic if and only if each of the* $\sqsubseteq_{i_j}$ *is b-monotonic.*

4.4 Truth Maintenance

In this section we shall employ the results developed above to give a semantical account of truth maintenance. While the definitions formulated and the results cited below can be extended to a very general notion of truth maintenance (including assumption-based truth maintenance) [Bro88], we shall restrict our attention to the classic nonmonotonic justification-based truth maintenance (JTMS) of Doyle [Doy79]. Our first task is to define the logical language $\mathcal{L}$ implicitly employed by truth maintenance. Let $\mathcal{P}$ be a collection of primitive propositions, of which p and q are typical members. Every primitive proposition is a well-formed formula (wff). If F_1 and F_2 are wff's, so are $\neg F_1$, $F_1 \leftarrow F_2$,[2] and $\Box F_1$. Formulae in the remaining standard Boolean connectives can be defined in the usual way from the ones already given. The standard *monotonic* semantics of $\mathcal{L}$ is given by any of the usual modal interpretations of modal propositional languages [GG84].[3] As usual, an interpretation satisfying every formula of a set of formulae is a *model* of that set. '$\Box$' is glossed 'it is believed that ...'. Formulae of the form $\Box F$ are *beliefs*, while those of the form $\neg\Box F$ are *negated* beliefs.

The language admitted by JTMS's is a restriction of the language described above. A formula F is *primitive* if and only if it is either a primitive proposition or negated primitive proposition. $\Box F$ is a *premiss* or *primitive belief* if F is primitive. A *justification* is any formula of the form $F \leftarrow F_1 \wedge \cdots \wedge F_n$ where F is a primitive belief and each F_i is either a primitive belief or negated primitive belief, F being the *consequent* of the justification and the F_i being the *antecedents*. Antecedents that are primitive beliefs are *monotonic* while those that are negated are *nonmonotonic*. A *JTMS theory* is any

[2] We use the leftward pointing arrow for the implication connective both to be consistent with our notation in related articles on this topic and because of the similarities between the logic of truth maintenance and that of logic programming.

[3] The choice of interpretation is largely a question of determining the degree to which propositions believed by a rational agent should be true propositions, and the degree to which this agent should be able to reflect upon its own beliefs. In [Bro88] we consider a class of interpretations that reflects *exactly* the choice made implicitly in various operational truth maintenance systems.

finite set of premisses and justifications. Note that a premiss is in effect a justification with no antecedents. A JTMS theory is termed nonmonotonic (monotonic) if it has (no) justifications with nonmonotonic antecedents. For readers familiar with Doyle's work, premisses and justifications in a JTMS theory are in correspondence with his homonymous notions. Primitive beliefs correspond to his 'nodes'.

We turn now to the *nonmonotonic* semantics of JTMS theories. Our aim is to characterize modal interpretations in such a way as to make the admissible models of JTMS theories satisfy exactly those primitive beliefs that a justification-based truth maintenance system would label as "IN". We shall employ the device of stratification introduced earlier.

Definition 4.4.1 *A modal interpretation* validates *a justification just in case one of the following holds:*

1. *the interpretation satisfies the consequent and all of the antecedents of the justification;*

2. *the interpretation fails to satisfy the consequent and at least one of the antecedents of the justification.*

Notice that an interpretation validates a premiss (a justification with no antecedents) if and only if the interpretation satisfies the premiss.

Definition 4.4.2 *A modal interpretation* validates *a primitive belief* $\Box F$ *under a set of justifications* S *just in case one of the following holds:*

1. $I \models \Box F$ *and there is a justification in* S *validated by* I *whose consequent is* $\Box F$,

2. $I \not\models \Box F$ *and every justification in* S *with consequent* $\Box F$ *is validated by* I.

The validation of a primitive belief corresponds to the intuition that whenever a primitive belief is satisfied in an interpretation, there ought to be some justification supporting that belief whose antecedents are also satisfied by the interpretation.

Definition 4.4.3 *Let* S *be a set of justifications, and* I_1 *and* I_2 *be modal interpretations of* $\mathcal{L}$. $I_2 \sqsubseteq_S I_1$ *if and only if every primitive belief validated by* I_2 *under* S *is validated by* I_1 *under* S. $I_2 \sqsubseteq_\perp I_1$ *if and only if every belief satisfied by* I_1 *is satisfied by* I_2.

A *justification graph* of a finite set of justifications is a directed graph containing a vertex for each justification and a directed edge from one vertex to another just in case the consequent of the justification corresponding to the first vertex is an antecedent or the negation of an antecedent of the justification corresponding to the second vertex. The justification graph can be partitioned into strongly connected components with the usual induced partial order on those components [AHU74]. Being a finite set of justifications, the partial order induced by partitioning the justification graph into strongly connected components is a *graded* partial order [Bir67] with the following grading:

1. minimal acyclic components in the partial order have grade 0;

2. minimal cyclic components in the partial order have grade 1;

3. the grade of a component that is not minimal is one greater than the upper bound among grades of its immediately antecedent (in the partial order) components.

The finite nature of the underlying set of justifications guarantees that there is a strongly connected component of largest (finite) grade. Notice that each premiss results in a strongly connected component of grade 0.

Definition 4.4.4 *Let $\mathcal{T}$ be a JTMS theory. Let $S_n \subseteq \mathcal{T}$ be those justifications whose corresponding vertices in the justification graph are in strongly connected components of grade n, where N is the maximum among grades of the strongly connected components of the justification graph of $\mathcal{T}$. A modal interpretation M of $\mathcal{L}$ is a* JTMS model *of a JTMS theory $\mathcal{T}$ just in case M is a modal model of $\mathcal{T}$, and there is no M' such that $M \sqsubseteq_{S_0} \sqsubseteq_{S_1} \cdots \sqsubseteq_{S_N} \sqsubseteq_{\perp} M'$ while it is not the case that $M' \sqsubseteq_{S_0} \sqsubseteq_{S_1} \cdots \sqsubseteq_{S_N} \sqsubseteq_{\perp} M$.*

To emphasize, a JTMS model is a modal interpretation that is at least as good as any other interpretation in the stratified ordering, *and* this maximal interpretation also happens to be a modal model of the JTMS theory. The stratified compounding of relations on interpretations allows us to encode in the semantics the idea of *well-foundedness* that plays a key role in truth maintenance.

Definition 4.4.5 *A modal interpretation M of a JTMS theory $\mathcal{T}$ is* well-founded *in $\mathcal{T}$ if and only if there is a partial order on primitive beliefs such that for every primitive belief satisfied by M there is a justification whose consequent is that belief, whose antecedents are satisfied by M, and all of whose monotonic antecedents precede the consequent belief in the partial order.*

The following proposition relates the (essentially) semantical notion of stratification to the (essentially) syntactic notion of well-foundedness

Proposition 4.4.1 *A modal interpretation is a JTMS model of $\mathcal{T}$ if and only if it is well-founded in $\mathcal{T}$ and is a modal model of $\mathcal{T}$.*

Notice that in definition 4.3.3 given above that an interpretation s-satisfied a set of formulae if it were a maximal (in the ordering on interpretations) interpretation satisfying that set of formulae. Here in contrast, we require that a JTMS model be maximal in the ordering *first* and then satisfy the set of formulae. If we reversed the order of maximization and satisfaction in the definition of JTMS models, theories $\mathcal{T}_2$ and $\mathcal{T}_5$ below would have JTMS models, which we do not want because the models in question would correspond to having ill-founded arguments for the beliefs satisfied.

Consider the following JTMS theories:

$$\begin{aligned}
\mathcal{T}_1 &= \{\Box p \leftarrow \Box p\},\\
\mathcal{T}_2 &= \{\Box p \leftarrow \neg\Box p\},\\
\mathcal{T}_3 &= \{\Box p \leftarrow \ulcorner q, \Box q \leftarrow \Box p\},\\
\mathcal{T}_4 &= \{\Box p \leftarrow \neg\Box q, \Box q \leftarrow \neg\Box p\},\\
\mathcal{T}_5 &= \{\Box p \leftarrow \neg\Box p, \Box p \leftarrow \Box q, \Box q \leftarrow \Box p, \Box q \leftarrow \neg\Box q\}.
\end{aligned}$$

The JTMS models of $\mathcal{T}_1$ are the modal interpretations satisfying no primitive beliefs.[4] $\mathcal{T}_2$ has no JTMS models. The JTMS models of $\mathcal{T}_3$ are the same as those of $\mathcal{T}_1$. $\mathcal{T}_4$ has two disjoint sets of JTMS models, those satisfying $\Box p$ and no other primitive belief and those satisfying $\Box q$ and no other primitive belief. $\mathcal{T}_5$ has no JTMS models. Notice that in this last case that the least modal interpretations are those validating $\Box p \leftarrow \Box q$ and $\Box q \leftarrow \Box p$, and satisfying no primitive modal beliefs. These interpretations are not, however, models of $\mathcal{T}_5$. These examples are illustrative of an important observation about truth maintenance: While one might imagine that truth maintenance is a computational realization of the *proof theory* of some logic, it appears, in fact, to be a realization of its model theory. This point of view is justified by the following observations:

1. A JTMS model can be constructed directly from a valid labelling provided by a justification-based truth maintenance system.

2. The failure of a justification-based truth maintenance system to produce a valid labelling is indicative of the nonexistence of JTMS models (though perhaps of a restricted class).

3. Construing a node's theoremhood from its being labelled "IN" requires the weakening of the usual definition of theorem, while taking it as being satisfied in some JTMS model of the theory requires no such change.

The following proposition formalizes the observation that viewed from the standard (monotonic) point of view a JTMS theory can only impose primitive beliefs and never negated primitive beliefs. As a consequence it is always consistent, hence has modal models. From the nonmonotonic point of view, however, negated primitive beliefs *can* be imposed hence admitting inconsistency with respect to that viewpoint.

Proposition 4.4.2 *Every JTMS theory has standard modal models but not necessarily JTMS models.*

The following two propositions capture the essential feature of nonmonotonicity, namely that the models for monotonic theories are in effect unique for each theory while nonmonotonic theories admit multiple incomparable models. It is nicely illustrated by $\mathcal{T}_2$ among the example JTMS theories above which has no JTMS models, but if we add $\Box p$ to the theory it does.

[4]Depending on the class of modal interpretations chosen, there may be non-primitive beliefs which are also satisfied.

Proposition 4.4.3 *Every monotonic JTMS theory has JTMS models and they satisfy precisely the same primitive beliefs.*

Proposition 4.4.4 *There are JTMS theories $\mathcal{T}_1 \subset \mathcal{T}_2$ such that no JTMS model of $\mathcal{T}_2$ is a JTMS model of $\mathcal{T}_1$.*

4.5 Conclusions

In the foregoing we have recapitulated our earlier results providing a framework for a semantical account of nonmonotonic reasoning. In the present work we have generalized from binary relations that are partial orders on interpretations to reflexive and transitive binary relations on interpretations. We passed thus from the intuitive notion of a "preference for" some interpretations to a "bias towards" some interpretations. We further extend the framework presenting a construction that compounds our biases towards interpretations. We have articulated the relation between the original formulation and its generalization and extension. While the particular version of stratification that we introduced was defined in terms of an *iterative* refinement of our biases, it turns out that the iterative construction is not essential. (Indeed, the idea of compounding can be extended in both constructive and non-constructive.) Finally, we make direct use of stratification to give a semantical account of justification-based truth maintenance systems.

Bibliography

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Massachusetts, 1974.

[Bir67] Garrett Birkhoff. *Lattice Theory*. Volume 25 of *American Mathematical Society Colloquium Publications*, American Mathematical Society, Providence, Rhode Island, third edition, 1967.

[Bro88] Allen L. Brown, Jr. Logics of justified belief. Manuscript submitted for publication.

[Doy79] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[GG84] Dov M. Gabbay and Franz Guenthner, editors. *Extensions of Classical Logic*. Volume 2 of *Handbook of Philosophical Logic*, D. Reidel, Dordrecht, Netherlands, 1984.

[Min87] Jack Minker. *Foundations of deductive databases and logic programming*. Morgan-Kauffmann, 1987.

[Sho86] Yoav Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. PhD thesis, Yale University, New Haven, Connecticut, 1986.

5. Logics of Justified Belief

Allen L. Brown, Jr.
GE Corporate Research and Development Center
P.O. Box 8
Schenectady, New York 12301
U.S.A.

Abstract

We give a formal semantics to truth maintenance by offering here a mathematical logic—equipped with an underlying model theory—that is used to characterise quite precisely some well known models of truth maintenance. Our usage of 'precise' is doubly intended in that we give meaning to truth maintenance in terms of a formal logic, *and* that each characterising logic corresponds to a particular truth maintenance system and *vice versa*.

5.1 Introduction

Although there are various logical accounts of nonmonotonic reasoning (see [Per84] for a complete survey) that have been equipped with suitable formal semantics (including our own attempt in [Bro85]), none of these accounts captures *both* nonmonotonic *and* assumption-based truth maintenance with satisfactory precision. The question we propose to answer is: With respect to what logic might the labelled formulae of truth maintenance systems be counted as theorems?

In the following we will develop logics and associated semantics that correspond to the the justification-based truth maintenance systems (JTMS's) of Doyle [Doy79] and Goodwin [Goo87], the assumption-based truth maintenance system (ATMS) of de Kleer [deK85], and our own algebraic nonmonotonic reason maintenance system (ANRMS) [BGB86]. We will first provide a logic and model theory for the ANRMS. We will then reduce the logical characterisation of other TMS's to the ANRMS case. As we pointed out earlier, our principle task is to make logical sense of truth maintenance labellings such as the "IN" and "OUT" of Doyle. We do this this by formalising the propositional attitude of *belief* for propositions relative to *standards* of credibility. We call these beliefs *justified* in that they are the consequents of syntactically well-formed arguments. We distinguish them from the *true beliefs* [Get67, Gri67, Mal67, Pri67] ordinarily of interest to philosophers in that we are disinterested in whether the propositions are rationally compatible (just as is the case in the fundamental computational mechanisms of truth maintenance systems). The logics we construct will give a syntactic and semantic characterisation to justifications and beliefs.

5.2 Syntax

Let $\mathcal{P}$ be a set of primitive propositions. p, q, r (possibly subscripted) will denote particular elements of $\mathcal{P}$. Let $\mathcal{L}$ be a Boolean lattice[1] with domain $\mathcal{D}$, meet ($\sqcap$), join ($\sqcup$) and complement ($^-$) operators, a partial order ($\sqsubset$), and least ($\bot$) and greatest ($\top$) elements. A, B (possibly subscripted) will denote particular elements of $\mathcal{L}$. Any element of $\mathcal{L}$ is a lattice expression, as are any meets, joins or complements of lattice expressions. x, y (possibly subscripted) will be syntactic variables ranging over lattice expressions. We define the well-formed formulae (wff's) of the multi-modal logical language as follows:

1. Every element of $\mathcal{P}$ is a wff;
2. If F is a wff, so is $\neg F$;
3. If F_1 and F_2 are wffs, so is $F_1 \leftarrow F_2$;
4. If F is a wff and $A \in \mathcal{L}$, then $[A]F$ is a wff.

The language $\mathcal{L}$ can be extended to include the other standard Boolean connectives by definition relative to '$\neg$' and '$\leftarrow$' in the usual way. Let $\widehat{\mathcal{L}}$ be an atomic sublattice of $\mathcal{L}$. For the purposes of the present exposition we wish to consider, a sublanguage of defined as follows.

1. If $p \in \mathcal{P}$ and $A \in \mathcal{L}$, $[A]\tilde{p}$, and $\neg[A]\tilde{p}$ all formulae of where $\tilde{p}$ denotes one of p or $\neg p$. '$[A]$' is a belief modality. Formulae of the form $[A]\tilde{p}$, and $\neg[A]\tilde{p}$ are called (respectively) *beliefs* and *negated* beliefs with *core* $\tilde{p}$. The sets of beliefs and negated beliefs will be denoted $[\mathcal{L}]\tilde{\mathcal{P}}$ and $\neg[\mathcal{L}]\tilde{\mathcal{P}}$ respectively. Similarly, if $P \subseteq \tilde{\mathcal{P}}$ ($\tilde{\mathcal{P}}$ being the set of possibly negated primitive propositions), $[\mathcal{L}]P$ and $\neg[\mathcal{L}]P$ are respectively the set of beliefs and negated beliefs whose cores are in P and whose modalities come from $\mathcal{L}$.

2. If $\tilde{p}, \tilde{q}_1, \ldots, \tilde{q}_m, \tilde{r}_1, \ldots, \tilde{r}_n$ are in $\tilde{\mathcal{P}}$ and A is an atom of $\widehat{\mathcal{L}}$, then

$$
\begin{aligned}
[A]\tilde{p} &\leftarrow [A]\tilde{q}_1 \wedge \cdots \wedge [A]\tilde{q}_m \\
[A]\tilde{p} &\leftarrow \neg[A]\tilde{r}_1 \wedge \cdots \wedge \neg[A]\tilde{r}_n \\
[A]\tilde{p} &\leftarrow [A]\tilde{q}_1 \wedge \cdots \wedge [A]\tilde{q}_m \wedge \neg[A]\tilde{r}_1 \wedge \cdots \wedge \neg[A]\tilde{r}_n
\end{aligned}
$$

   are all in . Formulae of the latter form are called *justifications* when $[A]\tilde{p}$ is the *consequent* of the justifications, while $[A]\tilde{q}_1, \ldots, [A]\tilde{q}_m$ and $\neg[A]\tilde{r}_1, \ldots, \neg[A]\tilde{r}_n$ are respectively the *monotonic* and *nonmonotonic antecedents* of the justifications where they are mentioned. Justifications without nonmonotonic antecedents are termed *monotonic* while those with are termed *nonmonotonic*.

[1] The logical results developed in this paper can be generalised to any complete latice. As we are interested in logically characterising *extant* truth maintenance systems, Boolean lattices are both sufficient and more direct in achieving our ends.

3. No other formulae are in$\hat{}$.

We presume the existence of a total lexical ordering on the formulae of .

For notational convenience we allow justification schemata that have the same form as justifications with lattice variables substituted for lattice elements. The intended interpretation of these schemata is that they stand for the justifications resulting from all possible substitution instances of atoms from $\hat{\mathcal{L}}$ for variables. Formally, a *TMS theory*, $\mathcal{T}$, is any *finite* set of beliefs, justifications and justification schemata having no explicit occurrences of $\bot$. The specific atomic lattice $\hat{\mathcal{L}}$ of interest to us is the sublattice of $\mathcal{L}$ generated by taking all the expressions in meets, joins and complements over the lattice elements appearing among the beliefs in $\mathcal{T}$. TMS theories are multi-modal propositional theories. Beliefs whose modality is an atom of $\hat{\mathcal{L}}$ will be termed *atomic*. We may think of a belief in a TMS theory as a justification having no antecedents. The set of *premisses* of a TMS theory are the beliefs it contains together with all the formulae $[B]\tilde{p}$ where $B = A_1 \sqcup \cdots \sqcup A_n$ and each of the A_i is an atom of $\hat{\mathcal{L}}$ such that there is a belief $[B']\tilde{p}$ in the TMS theory with $A_i \sqsubseteq B'$ for every i. It will be more convenient for us to think of a theory as being all of its premisses together with its justifications and all of the atomic instances of its justification schemata. A TMS theory having nonmonotonic justifications or justification schemata is nonmonotonic.

We interpret the modalities generated by the lattice as standards of credibility. With respect to that interpretation we give the following informal readings to formulae: $[A]p$ means that the proposition p is credible at the standard A. $\neg[B]q$ means that the proposition q is incredible at the standard B. $[A]r \leftarrow [A]p \wedge \neg[A]q$ means that should the proposition p be credible at the standard A and should the proposition q be incredible at the standard A, then the proposition r is credible at standard A.

$\mathcal{C}_\mathcal{T}$, the set of *completions* of $\mathcal{T}$, is the set of subsets, $\mathcal{S}$, of$\hat{}$ such that

1. $\mathcal{T} \subseteq \mathcal{S} \subseteq \hat{}$;
2. every formula $[\bot]\tilde{p}$ is in $\mathcal{S}$;
3. for every formula $\tilde{p} \in \tilde{\mathcal{P}}$ and every $A \in \hat{\mathcal{L}}$ either $[A]\tilde{p} \in \mathcal{S}$ or $\neg[A]\tilde{p} \in \mathcal{S}$;
4. no other formulae are in $\mathcal{S}$.

For $A, B \in \hat{\mathcal{L}}$ we define the *deduction operator* $\partial_\mathcal{T}$ for the theory $\mathcal{T}$ as follows:

1. $\mathcal{T} \subseteq \partial_\mathcal{T}(\mathcal{S})$;
2. $[\bot]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$;
3. $[A]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$ whenever $[B]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$ for $A \sqsubseteq B$;
4. $[A \sqcup B]\tilde{p}$ whenever $[A]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$ and $[B]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$;
5. $\neg[A]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$ whenever $\neg[B]\tilde{p} \in \partial_\mathcal{T}(\mathcal{S})$ for $A \sqsupseteq B$;

6. if $\neg[A]\tilde{p}$ and $[A]\tilde{p}$ are both in $\mathcal{S}$, then $\neg[B]\tilde{q}$ and $[B]\tilde{q}$ are in $\partial_{\mathcal{T}}(\mathcal{S})$ for every $\tilde{q}$ and B;

7. $[A]\tilde{p} \in \partial_{\mathcal{T}}(\mathcal{S})$ whenever there is a set of justifications in in $\mathcal{T}$ the cores of whose consequents are all $\tilde{p}$, the join of the belief modalities of the consequents is at least A, and all of whose antecedents are in $\mathcal{S}$;

8. $\neg[A]\tilde{p} \in \partial_{\mathcal{T}}(\mathcal{S})$ when for each justification in $\mathcal{T}$ with consequent $[A]\tilde{p}$, the negation of at least one of its antecedents is in $\mathcal{S}$;

9. $\neg[A]\tilde{p} \in \partial_{\mathcal{T}}(\mathcal{S})$ whenever $[A]\tilde{p} \notin \partial_{\mathcal{T}}(\mathcal{S})$;

10. no other formulae are in $\partial_{\mathcal{T}}(\mathcal{S})$.

We will say that $\mathcal{S}_1 \in \mathcal{C}_{\mathcal{T}} \lhd_{\mathcal{T}} \mathcal{S}_2 \in \mathcal{C}_{\mathcal{T}}$ if whenever a belief $[A]\tilde{p}$ is in the former set of formulae, it is also in the latter. We are typically interested in the least (with respect to $\lhd_{\mathcal{T}}$) fixed points of $\partial_{\mathcal{T}}$. We will refer to a fixed point of a theory, $\mathcal{T}$, meaning a fixed point of its deduction operator. A set of formulae, $\mathcal{S}$, will be termed *inconsistent* if it contains both a belief, $[A]\tilde{p}$, and its negation, $\neg[A]\tilde{p}$.

For least fixed points to be interesting they must exist:

Proposition 5.2.1 *Every TMS theory has a least fixed point.*

Since TMS labellings can obviously be nonmonotonic as, for example, when the addition of new justifications causes formulae formerly labelled as "IN" to be relabelled "OUT", the corresponding logical theory ought to have this property as well:

Proposition 5.2.2 *There exist TMS theories $\mathcal{T}_1$ and $\mathcal{T}_2$ such that there is no least fixed point of $\mathcal{T}_1 \cup \mathcal{T}_2$ containing any least fixed point of $\mathcal{T}_1$.*

A partial order, a subset of $\mathcal{D}^2$, is *graded* if there is a function from $\mathcal{D}$ into the non-negative integers such that

1. every $d \in \mathcal{D}$ has a grade;

2. the grade of $d \in \mathcal{D}$ is 0 whenever there is no $d' \in \mathcal{D}$ such that d' is less than d in the partial order;

3. the grade of each $d \in \mathcal{D}$ is larger than that of every $d' \in \mathcal{D}$ smaller than d in the partial order.

A fixed point, $\mathcal{S}$, of a theory, $\mathcal{T}$, is *well-founded* if there is a graded partial order, $<_{\mathcal{S}}$, on atomic beliefs (of $\mathcal{S}$) such that for every atomic belief $[A]\tilde{p} \in \mathcal{S}$, there is a justification whose consequent is $[A]\tilde{p}$, all of whose antecedents are in $\mathcal{S}$ and each of whose monotonic antecedents is less in the $<_{\mathcal{S}}$ ordering than $[A]\tilde{p}$.

We complete this section with some additional proof-theoretic results for TMS theories that will serve us later in our investigation.

Proposition 5.2.3 *Let $\mathcal{S}$ be a well-founded least fixed point of the TMS theory $\mathcal{T}$. Let $<_S$ be a graded partial order for $\mathcal{S}$. There exists a least partial order contained in $<_S$ under which $\mathcal{S}$ remains well-founded.*

A TMS theory, $\mathcal{T}$, together with $\partial_{\mathcal{T}}$ is a logic of justified belief. As mentioned earlier, we speak of justified beliefs rather than true beliefs. For a belief to be justified we merely require it to be grounded in a well-founded argument (the partial order on a fixed point extended to include justifications). Thus it is possible for both $[A]p$ and $[A]\neg p$ to be justified in a *consistent* TMS theory even though this pair of beliefs ought not be held by a *rational* agent. This contrasts with logics of true belief wherein the cores of positive (negative) beliefs are typically (non-)theorems is some underlying non-modal theory. In general we shall be interested in least fixed points of the justification operators of particular TMS theories, where those fixed points are well-founded under the associated partial order. Consider the TMS theories over the lattice $\{\bot, \top\}$

$$\begin{aligned}
\mathcal{T}_1 &= \{[\top]p \leftarrow \neg[\top]p\},\\
\mathcal{T}_2 &= \{[\top]p \leftarrow [\top]q, [\top]q \leftarrow [\top]p\},\\
\mathcal{T}_3 &= \{[\top]p \leftarrow \neg[\top]q, [\top]q \leftarrow \neg[\top]p\},\\
\mathcal{T}_4 &= \{[\top]p \leftarrow \neg[\top]p, [\top]p \leftarrow [\top]q, [\top]q \leftarrow [\top]p, [\top]q \leftarrow \neg[\top]q\}.
\end{aligned}$$

$\mathcal{T}_1$ has a single least fixed point, $\mathcal{T}_1 \cup [\widehat{\mathcal{L}}]\tilde{\mathcal{P}} \cup \neg[\widehat{\mathcal{L}}]\tilde{\mathcal{P}}$, and it is inconsistent. $\mathcal{T}_2$ has two consistent least fixed points, $\mathcal{T}_2 \cup [\bot]\tilde{\mathcal{P}} \cup \neg[\top]\tilde{\mathcal{P}}$ and $\mathcal{T}_2 \cup [\bot]\tilde{\mathcal{P}} \cup \neg[\top](\tilde{\mathcal{P}} - \{p, q\}) \cup \{[\top]p, [\top]q\}$ of which the first is well-founded. $\mathcal{T}_3$ has two least fixed points, $\mathcal{T}_3 \cup [\bot]\tilde{\mathcal{P}} \cup \{[\top]p\} \cup \neg[\top](\tilde{\mathcal{P}} - \{p\})$ and $\mathcal{T}_3 \cup [\bot]\tilde{\mathcal{P}} \cup \{[\top]q\} \cup \neg[\top](\tilde{\mathcal{P}} - \{q\})$, and each of them is consistent and well-founded. $\mathcal{T}_4$ has a single least fixed point, $\mathcal{T}_4 \cup [\bot]\tilde{\mathcal{P}} \cup \{[\top]p, [\top]q\} \cup \neg[\top](\tilde{\mathcal{P}} - \{p, q\})$ and it is consistent and *not* well-founded.

Proposition 5.2.4 *If $\mathcal{T}$ is a monotonic TMS theory, then it has a unique consistent well-founded least fixed point.*

The deduction operator is meant to capture the constraint propagation processes implicit in the various truth maintenance systems. The fixed points of TMS theories meant to capture that which has been *proven* in some underlying deductive theory in contrast to that which is *provable*. The correspondence between the syntactic notion of justification given above and the homonymous notion in the TMS's of Doyle and others will be apparent to readers familiar with those investigators' systems. Our aim here is for the justifications in $\mathcal{T}$, having no antecedents, to correspond to the *premisses* of a typical truth maintenance system. We mean for *consistent* completions and truth maintenance labellings to have the following correspondence: For a given consistent completion containing the belief $[A]\tilde{p}$ and *not* containing $[B]\tilde{p}$ for $A \sqsubset B$ (*i.e.* A is a maximal standard at which $\tilde{p}$ is believed), the corresponding truth maintenance labelling would have the label A on the node identified with $\tilde{p}$. Readers familiar with Doyle's JTMS may readily verify that a TMS node identified with $\tilde{p}$ being labelled "IN" correlates with $[\top]\tilde{p}$ being

in the corresponding consistent completion. Having given a syntactic characterisation to truth maintenance by defining a formal logic of justified belief, we turn now to supplying a suitable semantics for that logic.

5.3 Semantics

In this section we will equip TMS logics with a possible world semantics [Che80, GG84, HC68]. Specifically, we make use of neighbourhood interpretations[2] [GG84]. In order to capture the nonmonotonicity of TMS theories, we base our semantics on the idea of *minimal* models, a notion introduced by McCarthy [McC80] and Davis [Dav80], further pursued by Bossu and Siegel [BS85], and ultimately explored and exploited by Shoham [Sho86]. Finally, it will develop that the computation carried out by a truth maintenance system will correspond to the construction of an appropriate model should such a structure exist.

A *neighbourhood interpretation*, $\mathcal{I}$, is a structure $\langle W, \pi, \mu \rangle$ where

1. $\mathcal{W}$ is a non-empty set of worlds;
2. $\pi: \mathcal{P} \rightarrow 2^{\mathcal{W}}$, the range being the set of subsets of $\mathcal{W}$;
3. $\mu: \widehat{\mathcal{L}} \rightarrow 2^{(2^{\mathcal{W}})^{\mathcal{W}}}$, the range being the set of subsets of the functions from $\mathcal{W}$ to subsets of $\mathcal{W}$
4. if $B = \bigsqcup_{1 \leq i \leq n} A_i$ where the $A_i \in \mathcal{L}$, then $\mu(B) = \bigcap_{1 \leq i \leq n} \mu(A_i)$;
5. for every p, w there is an $f \in \mu(\bot)$ such that $f(w) = \pi(p)$;
6. for every $\neg p, w$ there is an $f \in \mu(\bot)$ such that $f(w) = \mathcal{W} - \pi(p)$.

We will subscript the various elements of a structure as required to avoid ambiguity of reference. Let F, F_1 and F_2 be formulae of . An interpretation $\mathcal{I}$ *satisfying* ($\models$) a formula at a world is defined by the following cases

1. $\mathcal{I}, w \models p \in \mathcal{P}$ if $w \in \pi(p)$;
2. $\mathcal{I}, w \models \neg F$ if $\mathcal{I}, w \not\models F$;
3. $\mathcal{I}, w \models F_1 \leftarrow F_2$ if either $\mathcal{I}, w \models F_1$ or $\mathcal{I}, w \not\models F_2$;
4. $\mathcal{I}, w \models [A]F$ if there is a function $f \in \mu(A)$ such that $f(w) = \{w' \in \mathcal{W} | \mathcal{I}, w' \models F\}$.

[2]Chellas [Che80] calls these *minimal* models, but we wish to reserve that term for another usage. By analogy with dynamic logic [GG84] where every programme induces a modality and an accessibility relation, in TMS logic each lattice expression (standard of belief) over $\widehat{\mathcal{L}}$ induces a modality and a family of accessibility functions.

A neighbourhood interpretation satisfies a formula if it satisfies it at every world. A neighbourhood interpretation is a neighbourhood *model* of a set of formulae (including a TMS theory) if it satisfies all of them. $\mathcal{I}_1 \prec_{\mathcal{T}} \mathcal{I}_2$ for neighbourhood interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ if $\mathcal{I}_2$ satisfies every belief satisfied by $\mathcal{I}_2$.

We need a semantic characterisation of well-foundedness.[3] To achieve this we define a least neighbourhood model $\mathcal{M}$ of a TMS theory $\mathcal{T}$ to be *forthright* if there is a sequence of neighbourhood models $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n = \mathcal{M}$, and a sequence of subsets of $\mathcal{T}$, $S_0 \subset S_1 \subset \cdots \subset S_n$ such that

1. S_0 is the set of premisses of $\mathcal{T}$;
2. S_n contains S_0 together with a subset of the justifications of $\mathcal{T}$;
3. $\mathcal{M}_i \models S_i$;
4. $\mathcal{M}_{i-1}$ satisfies the antecedents of each of the justifications in S_i but none of the consequents of $S_i - S_{i-1}$;
5. $\mathcal{M}_i$ satisfies the antecedents and consequents of each of the justifications in S_i;
6. $\mathcal{W}_i = \mathcal{W}_j$ and $\pi_i = \pi_j$ for $i, j \leq n$;
7. $\mu_i(A) \subset \mu_j(A)$ for $i \leq j$ and all $A \in \widehat{\mathcal{L}}$;
8. if $\mathcal{M}_i \models [B]\tilde{q}$, then $\mathcal{M}_j \models [B]\tilde{q}$ for $i < j$;
9. if $\mathcal{M}_{i-1} \not\models [B]\tilde{q}$ and $\mathcal{M}_i \models [B]\tilde{q}$, then there is a set of justifications in S_i whose consequents are $[B_1]\tilde{q}_1, \ldots, [B_m]\tilde{q}_m$ with $B \sqsubseteq \bigsqcup_{1 \leq i \leq m} B_i$.

A *TMS model* of $\mathcal{T}$ is a forthright, minimal (in the partial order $\prec_{\mathcal{T}}$) neighbourhood model of $\mathcal{T}$. Intuitively, a TMS model says that as few beliefs as possible are satisfied and that the antecedency relation is a partial order with respect to beliefs. The sequence S_i captures the argument supporting any particular belief, and such beliefs are ultimately grounded either in premisses or in negated beliefs. Indeed, the sequence of subsets necessary to forthrightness can be viewed as inducing a second partial order on neighbourhood models. TMS models can then be viewed as minimal with respect to both partial orders. We complete our study of TMS semantics with a completeness result for TMS theories:

Theorem 5.3.1 *Let $\mathcal{T}$ be a TMS theory. Let $S \in C_{\mathcal{T}}$ be consistent. $\mathcal{M}$ is a TMS model of S and $\mathcal{T}$ if and only if S is a least, well-founded fixed point of $\partial_{\mathcal{T}}$.*

[3] In [SB88] we achieve this end by means *stratified entailment*, an abstraction closely related to the technical device of *stratification* that has been intensively studied in the logic programming community. Unfortunately our limited space here obliges to rely on a less scenic but more direct path.

5.4 Particular Truth Maintenance Systems

Logics of justified belief in their full generality have been conceived to give a logical account of lattice-based truth maintenance. Since other well-known forms of truth maintenance are specialisations of lattice-based truth maintenance, their logical accounts are given by analogous specialisations of logics of justified belief. We will borrow liberally from [BBG86] wherein we elaborate the reductions of justification-based truth maintenance and assumption-based truth maintenance to our own model of truth maintenance based on Boolean lattices. We briefly sketch the logical accounts of Doyle's JTMS and de Kleer's ATMS based on those reductions.

Recall from [BGB86] that a lattice-equational system is a set of equations, each having either the form $s_k = A_k$ or the form $s_k = \bigsqcup_{j \in J_k} \bigsqcup_{i \in I_{j,k}} \tilde{s}_i$ where $A_k \neq \bot$ is an element of the Boolean lattice $\mathcal{L}$ and the s's are unknowns ranging over lattice elements, with each unknown standing for (a possibly negated) primitive proposition. The corresponding TMS theory is formed in the following way: For each unknown, s_k, there is a corresponding proposition, $\tilde{p}_k$.[4] For each disjunct of each equation there is a justification in the sense defined in section 5.2. For example, an equation of the first form above yields the premiss $[A_k]\tilde{p}_k$. A disjunct, say $\bigsqcup_{i \in I_{j,k}} \tilde{s}_i$, of an equation of the second form yields the justification schema, $[x]p_k \leftarrow \bigwedge_{i \in I_{j,k}} \{\neg\}[x]\tilde{p}_i$ where the optional negation ($\{\neg\}$) occurs on the right-hand side whenever the corresponding unknown was complemented in the original lattice equation. Recalling from [BGB86] that a set of JTMS justifications is rendered as a lattice-equational system over the lattice $\{\bot, \top\}$, the logical account of the JTMS follows immediately.

Capturing the assumption-based truth maintenance of de Kleer requires some additional elaboration. Let $\mathcal{A}$ be a finite set of assumptions. The set of sets $2^{2^{\mathcal{A}}}$ forms a Boolean lattice with set containment, intersection and union playing the roles of partial order, meet and join. Lattice complementation is the set complement of an element of the lattice with respect to the maximal element $2^{\mathcal{A}}$. It is in this lattice that assumption-based truth maintenance implicitly operates, though no use is made of set complementation. We define *widening* and *narrowing* of elements of $2^{2^{\mathcal{A}}}$ as follows: $x \Uparrow = \{y \in 2^{\mathcal{A}} | \exists z \in x[z \subseteq y]\}$ and $x \Downarrow = \{y \in x | \neg \exists z \in x[z \subset y]\}$. A system of ATMS justifications is straightforwardly translatable into a lattice-equational system where the labels on premiss nodes become widenings of the ATMS labellings. From lattice equational systems we pass to TMS theories by the same recipe as cited above. The solution to the lattice-equational system after narrowing is exactly the same as the labelling that the ATMS would have produced.

[4] Keep in mind that primitive propositions marked with '~' denote possibly negated primitive propositions, while similarly marked lattice unknowns denote possibly complemented unknowns.

5.5 Relation to Other Logical Systems

While the principal aim of this work is to give a "tight" (*i.e.* no more, no less) logical characterisation of truth maintenance, logics of justified belief exhibit noteworthy similarities in terms of both specific attributes and overall formal structure to a number of other logical systems. We sketch briefly below the connections with some of those other systems.

In [ABW87] Apt, Blair and Walker introduce the idea of a *stratified* logic programme which has been further elaborated and exploited by various investigators. By disallowing certain combinations of recursion and negation, a logic programme can be given a simple declarative and procedural meaning, the latter being equivalent to a complete proof procedure. In operational truth maintenance systems there are similar restrictions on combining negation and recursion (*e.g.* the prohibition of so-called "odd loops"), the effect being to reduce the computational complexity of calculating truth maintenance labellings. Stratification is very closely related to the concepts of well-foundedness and forthrightness that we have defined in this paper. The essence of both stratification and well-foundedness is that certain partial orders are induced that can be exploited in the construction of standard models for the respective logical theories.

In [Lev84], [FH85] and [HM85] Fagin, Halpern, Levesque, and Moses explore logics whose intention is to be expressive of the concepts of implicit and explicit belief, awareness, knowledge and limited reasoning. Of course, one of the original aims of truth maintenance systems was to formalise limited reasoning. Logics of justified belief give a logical account of that formalisation. Indeed, the limited logical language presented here *only* allows explicit belief, although there is an obvious extension to include implicit beliefs. Another point of similarity with the work of the authors cited is that irrational beliefs may be entertained by their logical systems without introducing logical inconsistency. Finally, the ideas of *common* knowledge and *implicit* knowledge among reasoning agents is closely related to the meet and join operations on the lattice structure that we have made use of in this work.

In [Gin87] and [Fit87] Fitting and Ginsberg explore the idea of interpreting logics over a general bilattice (in contrast to the usual two-valued Boolean lattice). In so doing they achieve plausible logical characterisations of evidential and default reasoning and truth maintenance. It should first be noted that there is an easy translation from (non-modal) logics interpreted over a lattice of truth values to multi-modal logics interpreted over the usual truth values, with the modalities having a lattice structure. While Ginsberg's characterisation of truth maintenance has strong underlying similarities to our own, we feel that our modal approach more directly captures the idea of relative belief implicit in truth maintenance. Also, in its full generality our system allows the ascription of belief to compound structures to be independent of the belief ascribed to the constituents.

In [RdK87] Reiter and de Kleer endeavour to give a precise logical characterisation to de Kleer's ATMS. They succeed admirably as well as generalising the idea of a truth maintenance system to *clause maintenance*. The structure of prime implicants that they

elucidate for the ATMS can be (and is) exploited in the implementation of a lattice-based truth maintenance system. There are three technical differences between their work and ours that we should like to highlight. Unlike their logical account, assumptions in our logic (the lattice of modalities) are ontologically distinct from primitive propositions. This is reflected in both their syntax and semantics. While some may see this as merely a matter of taste, we believe that our realisation of assumptions better reflects the intentionality of assumptions as "mental states" with respect to which various things might be believed. The other two differences are of a more substantive technical nature: First, our logic gives an account of nonmonotonicity. (Of course, the technical device of minimal models could be used to realise a nonmonotonic version of Reiter and de Kleer's logic too!) Second, that their logic really does not distinguish assumptions from propositions prohibits the possibility of holding both a primitive proposition and its negation to believed relative to the same assumption. While it is often the case that a problem solver wishes to impose exactly such a prohibition, we have encountered a number of applications in which this prohibition is *not* desirable.

In [San87] Sandewall attempts to address the defects of partial models as semantical accounts of partial knowledge. The essential feature of his approach is a truth valuation that assigns every a truth value in a four-valued lattice. This permits the "knowing" the truth value of a compound proposition while being ignorant of the truth values of its constituents. Logics of justified belief can express that epistemic state, as well as some states that are not expressible in Sandewall's semantics. While there seems to be a certain inter-expressibility between his and our logic, much more interesting is the correspondence between minimal interpretations we have defined among the partially ordered neighbourhood interpretations of TMS theories and the *condensations* among the sets of his epistemic interpretations that he defines. A TMS model "believes" as little as possible while a condensation "knows" as little as possible.

5.6 Conclusions

We have defined a collection of logical theories and associated them with various models of truth maintenance. We have identified the truth maintenance concepts of premiss, assumption, justification, node, "IN" and "OUT" with certain syntactic constructs in those logics. We have characterised the proof theories of those logics in terms of a deduction operator and its fixed points from the set of completions of those theories. Each instance of a given logic is uniquely identified with a set of premisses and justifications in a corresponding truth maintenance paradigm (and *vice versa*). We have given a semantical account of these logics in terms of minimal models. As it turns out, the labelling process carried out by a truth maintenance system corresponds to the construction of a minimal model should such an object exist.

Bibliography

[ABW87] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of deductive databases and logic programming*, Morgan-Kauffmann, 1987.

[BBG86] Dan Benanav, Allen L. Brown, Jr., and Dale E. Gaucas. Reason maintenance from a lattice-theoretic point of view. In Lee S. Baumann, editor, *Proceedings of the Expert Systems Workshop*, pages 83–87, Defense Advanced Research Projects Agency, Science Applications International Corporation, Pacific Grove, California, April 1986.

[BGB86] Allen L. Brown, Jr., Dale E. Gaucas, and Dan Benanav. An algebraic foundation for truth maintenance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 973–980, Milano, 1987.

[Bro85] Allen L. Brown, Jr. Modal propositional semantics for reason maintenance systems. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 178–184, Los Angeles, 1985.

[BS85] Geneviève Bossu and Pierre Siegel. Saturation, non-monotonic reasoning, and the closed world assumption. *Artificial Intelligence*, 25(1):13–64, January 1985.

[Che80] Brian F. Chellas. *Modal Logic: an Introduction.* Cambridge University Press, London, 1980.

[Dav80] M. Davis. The mathematics of non-monotonic reasoning. *Artificial Intelligence*, 13:73–80, 1980.

[deK85] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[Doy79] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[FH85] Ronald Fagin and Joseph Y. Halpern. Belief, awareness, and limited reasoning: a preliminary report. In *Proceedings of the Ninth International Joint*

Conference on Artificial Intelligence, pages 491–501, Los Angeles, August 1985.

[Fit87] Melvin Fitting. Logic programming on a topological bilattice. Unpublished manuscript.

[Get67] Edmund L. Gettier. Is justified true belief knowledge? In A. Phillips Griffiths, editor, *Knowledge and Belief*, chapter X, Oxford University Press, 1967.

[GG84] Dov M. Gabbay and Franz Guenthner, editors. *Extensions of Classical Logic*. Volume 2 of *Handbook of Philosophical Logic*, D. Reidel, Dordrecht, Netherlands, 1984.

[Gin87] Matthew L. Ginsberg. Multi-valued logics. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 243–247, Seattle, 1987.

[Goo87] James W. Goodwin. *A Theory and System for Non-Monotonic Reasoning*. PhD thesis, Linköping University, Linköping, Sweden, May 1987.

[Gri67] A. Phillips Griffiths. On belief. In A. Phillips Griffiths, editor, *Knowledge and Belief*, chapter IX, Oxford University Press, 1967.

[HC68] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic*. Methuen, London, 1968.

[HM85] Joseph Y. Halpern and Yoram Moses. A guide to the modal logics of knowledge and belief: preliminary draft. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 480–490, Los Angeles, August 1985.

[Lev84] Hector Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 198–202, Austin, 1984.

[Mal67] Norman Malcolm. Knowledge and belief. In A. Phillips Griffiths, editor, *Knowledge and Belief*, chapter V, Oxford University Press, 1967.

[McC80] John McCarthy. Circumscription: a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Per84] Donald Perlis. Bibliography of literature of non-monotonic reasoning. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 396–401, New Paltz, New York, October 1984.

[Pri67] H.A. Prichard. Knowing and believing. In A. Phillips Griffiths, editor, *Knowledge and Belief*, chapter IV, Oxford University Press, 1967.

[RdK87] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188, Seattle, 1987.

[San87] Erik Sandewall. Semantic structures. Unpublished manuscript.

[SB88] Yoav Shoham and Allen L. Brown, Jr. New results on semantical nonmonotonic reasoning. Manuscript submitted for publication.

[Sho86] Yoav Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. PhD thesis, Yale University, New Haven, Connecticut, 1986.

6. Uncertainty and Incompleteness: Breaking the Symmetry of Defeasible Reasoning

Piero P. Bonissone David A. Cyrluk James W. Goodwin* Jonathan Stillman

Artificial Intelligence Program
General Electric Corporate Research and Development
Schenectady, New York 12301
Arpanet: bonissone@ge-crd.arpa cyrluk@ge-crd.arpa stillman@ge-crd.arpa

Abstract

Two major difficulties in using default logics are their intractability and the problem of selecting among multiple extensions. We propose an approach to these problems based on integrating nonmonotonic reasoning with plausible reasoning based on triangular norms. A previously proposed system for reasoning with uncertainty (RUM) performs uncertain monotonic inferences on an acyclic graph. We have extended RUM to allow nonmonotonic inferences and cycles within nonmonotonic rules. By restricting the size and *complexity* of the nonmonotonic cycles we can still perform efficient inferences. The uncertainty measures in RUM provide a basis for deciding between multiple defaults. Different algorithms and heuristics for finding the optimal defaults are discussed.

6.1 Introduction

The management of uncertain information in first generation expert systems, when addressed at all, has largely been left to *ad hoc* methods. This has been effective only because operational expert systems normally assume that knowledge is complete, precise, and unvarying. This fundamental assumption is a principal source of the limitation of many diagnostic systems to single fault diagnoses, and the limitation of classification systems to time-invariant phenomena. See references [2] and [Pea88] for a survey of approaches to reasoning with uncertainty.

The management of incomplete information has also lacked a clear focus, as some researchers have attempted to find its solution by defining new nonmonotonic logics, by augmenting classical logic with default rules of inference, by searching for minimal models via functional optimization, or by concentrating only on the instruments, i.e.

*Currently with Knowledge Analysis, Belmont, Massachusetts.

TMSs, rather than the theory required to handle this problem. See references [BB85], [Rei88] and [Eth88] for a survey of approaches to reasoning with incompleteness.

In the past, a subset of the authors have contributed to the development of individual theories for reasoning with uncertainty and incompleteness. Bonissone has proposed RUM, a system for reasoning with uncertainty whose underlying theory is anchored on the semantics of many-valued logics [3]. This system provides a representation layer to capture structural and numerical information about the uncertainty, an inference layer to provide a selection of truth-functional triangular-norm based calculi [1], and a control layer to focus the reasoning on subsets of the KB, to (procedurally) resolve ignorance and conflict, and to maintain the integrity of the inference base via a belief revision system. RUM, however, does not provide any declarative representation to handle incomplete information.

Goodwin [Goo87] and Brown [BGB87] have provided such a representation by developing theories based on nonmonotonic dependency networks and algebraic equations over boolean lattices, respectively. These approaches, however, have totally neglected the aspect of uncertain information.

Another motivation is the existence of a new class of problems, referred to as *dynamic classification problems* [BW88], which cannot be properly addressed without an integration of the theories for reasoning with uncertainty and incompleteness. Preliminary work in this integration have been reported by D'Ambrosio (integrating assumptions and probabilistic reasoning) [Dam88] and Brown (integrating assumptions and nonmonotonic justification with uncertainty measures) [BBS88].

6.1.1 Proposed Approach

We have concentrated our efforts in integrating defeasible reasoning (based on nonmonotonic rules) with plausible reasoning (based on monotonic rules with partial degrees of sufficiency and necessity). In this paper we will present the preliminary results of such an integration.

In our approach, uncertainty measures are propagated through a Doyle-JTMS graph, whose labels are real-valued certainty measures.[1] Unlike other default reasoning languages that only model the incompleteness of the information, our approach uses the presence of numerical certainty values to distinguish *quantitatively* the different admissible labelings and pick an *optimal* one.

The key idea is to exploit the information on the monotonic links carrying uncertainty measures. A preference function based on such measures is used to select the extension, i.e., the fixed point of the nonmonotonic loop, which is maximally consistent with the soft constraints imposed by the monotonic links. Thus, instead of minimizing the cardinality of abnormality types [McC86] or of performing temporal minimizations [Sho86],

[1] It is also possible to decompose such a graph into a directed acyclic graph (DAG), whose nodes can either be object-level variables or nonmonotonic loops. The links in the DAG are plausible inference rules with Horn clause restrictions. The nonmonotonic loops are Strongly Connected Components (SCCs) of the graph, containing nonmonotonically justified rules.

we maximize an expectation function based on the uncertainty measure. This method breaks the symmetry of the (potentially) multiple extensions in each loop by selecting a *most likely* extension. This idea is currently being implemented in PRIMO (Plausible ReasonIng MOdule), RUM's successor.

The following section defines PRIMO's rule-graph semantics and constraints. Section 3 describes the generation of admissible labelings (consistent extensions) and introduces an objective function to guide the selection of preferred extensions. A small example illustrating the algorithm for propagating bounds through a PRIMO graph is shown in Section 4. Section 5 deals with optimization techniques (applicable on restricted classes of graphs) and heuristics (such as graph decomposition into strongly connected components), which can be used to generate acceptable approximation to the optimal solution. The conclusion section summarizes our results and defines an agenda of possible future research work.

6.2 *P*lausible *R*eason*I*ng *MO*dule

The decision procedure for a logic based on real-valued truth values may be much more computationally expensive than that for boolean-valued logic. This is because in boolean-valued logic only one proof need be found. In real-valued logic all possible proofs must be explored in order to ensure that the certainty of a proposition has been maximized.

RUM (*Reasoning with Uncertainty Module*), *the predecessor to* PRIMO, was designed as a monotonic expert system shell that handles uncertainty according to triangular norm calculi[2]. It deals with the possible computational explosion by allowing only propositional acyclic[3] quantitative Horn clauses.

To avoid the problems of first order reasoning, RUM restricts it rules to be propositional. RUM allows the user to write first-order rules, but insists that they are fully instantiated at run time. Thus a single rule may give rise to many rules at run time, all of which are propositional, thus avoiding the problems of first-order reasoning.

RUM restricts its rules to Horn clauses; it deals with negative antecedents by treating P and $\neg P$ independently. We denote *the certainty of* P as LB(P). The only time P and $\neg P$ will interact is when LB(P) + LB($\neg P$) > 1 (both P and $\neg P$ are believed). When this occurs a conflict handler tries to detect the source of inconsistency[4].

Due to these restrictions a simple linear time algorithm exists for propagating certainty values through RUM rules. Resolution of inconsistency by the conflict handler, however, may require cost exponential in some subset of the rules.

PRIMO (Plausible ReasonIng MOdule) is the successor to RUM designed to perform nonmonotonic reasoning. PRIMO extends RUM by allowing nonmonotonic antecedents.

[2]Triangular norm calculi represent logical *and* as a real valued function called a t-norm, and logical *or* as a s-conorm. For an introduction to them see [3].

[3]Unless an idempotent t-norm is used cyclic rules will cause all certainties in the cycle to converge to 0.

[4]Note that the above constraint on LBs implies an upper-bound on LB(P) of $1 - \text{LB}(\neg P)$. In the literature this is denoted as UB(P). LB and UB are related just as support and plausibility in Dempster-Shafer, or $\Box$ and $\Diamond$ in modal logics.

PRIMO also allows nonmonotonic cycles which represent conflicts between different defaults. We provide a formal overview of PRIMO below:

Definitions: A PRIMO specification is a triple (L, I, J). L is a set of ground literals. such that whenever $l \in L$, $\bar{l} \in L$. For $l \in L$, LB(l) $\in [0, 1]$ is the amount of evidence confirming the truth of l. J is a set of justifications. A justification, j, is of the form:

$$\bigwedge_i ma_i \wedge \bigwedge_i nma_i \rightarrow^s c$$

where $s \in [0, 1]$, the *sufficiency* of the justification, indicates the confidence of the justification; $ma_i \in L$, are the monotonic antecedents of j; nma_i are the nonmonotonic antecedents of j, and have the form, $\neg\boxed{\alpha}p$, where $p \in L$, with the semantics:

$$LB(\neg\boxed{\alpha}p) = \begin{cases} 0 & \text{if LB}(p) \geq \alpha \\ 1 & \text{if LB}(p) < \alpha \end{cases}$$

The *input literals* $I \subset L$, are a distinguished set of ground literals for which a certainty may be provided by outside sources (e.g. user input), as well as by justifications. The certainty of all other literals can only be affected by justifications.

A PRIMO specification can also be viewed as an AND/OR graph, with justifications mapped onto AND nodes and literals mapped onto OR nodes.

Definition: A valid PRIMO graph is a PRIMO graph that does not contain any cycles consisting of only monotonic edges.

Definition: An *admissible labeling* of a PRIMO graph is an assignment of real numbers in [0, 1] to the arcs and nodes that satisfy the following conditions:

1. the label of each arc leaving a justification equals the t-norm of the arcs entering the justification and the sufficiency of the justification and

2. the label of each literal is the s-co-norm of the labels of the arcs entering it.

A PRIMO graph may have zero, one, or many admissible labelings. An odd loop (a cycle traversing an odd number of nonmonotonic wires) is a necessary but not sufficient condition for a graph to have no solutions. Every even cyclic graph has at least two solutions. In these respects PRIMO is like the Doyle JTMS [Doy79]. Proofs can be found in [Goo88]

6.3 Finding Admissible Labelings

As with most TMS problems, it is natural to propagate constraints as far as possible before resorting to search.

6.3.1 Propagation of Bounds (PB)

In PRIMO, propagation of bounds (PB) on LB's will penetrate further than propagation of exact values alone. It may even trigger further propagation of exact values when bounds are propagated to a nonmonotonic antecedent whose value of α falls outside of them. Thus PB sometimes solves an entire cycle exactly which is impenetrable to propagation of exact values alone.

PB labels vertices with pairs of values representing lower and upper bounds on the exact LB of that vertex in any admissible labeling. These bounds are successively narrowed as propagation continues. For a vertex v at any time during execution, we define $\mathrm{LB}^-(v)$ and $\mathrm{LB}^+(v)$, the lower and upper bounds on $\mathrm{LB}(v)$ at that time, to be functions of the bounds then stored on the antecedents of v. LB^- uses the lower bounds of monotonic antecedents and the upper bounds of nonmonotonic ones; LB^+ uses the upper bound of monotonic and the lower bound of nonmonotonic antecedents. The actual function applied to these values is the same one used to compute LB itself for that vertex. The algorithm is then:

1. Initialize every input node, where k is the confidence given by the user, to $[k, 1]$ i.e. "at least k". Initialize every other vertex to $[0, 1]$.

2. While there exists any vertex v such that the label on v is not equal to $[\mathrm{LB}^-(v), \mathrm{LB}^+(v)]$, relabel v with that value.

It can be shown that PB converges in polynomial time, yields the same result regardless of the order of propagation, and never assigns bounds to a vertex which exclude any value that vertex takes on in any admissible labeling. (Thus PB will never find an exact solution for a graph which has more than one.) Proofs can be found in [Goo88].

6.3.2 A Labeling Algorithm for PRIMO

Definitions: A nonmonotonic antecedent is *satisfied* if $\mathrm{LB}^+ < \alpha$, *exceeded* if $\mathrm{LB}^- \geq \alpha$, and *ambiguous* if $\mathrm{LB}^- < \alpha \leq \mathrm{LB}^+$. A labeled graph is *stable* if it is closed under PB, i.e., every vertex v is labeled $[\mathrm{LB}^-(v), \mathrm{LB}^+(v)]$. In a stable graph, a *starter dependency* is an AND-vertex which has no unlabeled monotonic antecedents, no exceeded nonmonotonic antecedents, and at least one ambiguous nonmonotonic antecedent.

A starter dependency must be unlabeled, with a zero LB^- and a positive LB^+. Because PRIMO nets contain no monotonic loops, a starter dependency always exists (unless PB labeled the entire graph exactly) and can be found in time linear in the size of the graph. Because the only inputs left undetermined are nonmonotonic antecedents (i.e., thresholds) a starter dependency must be labeled exactly LB^- or LB^+ in any admissible labeling which may exist [Goo88].

One can therefore find all admissible labelings of a stable graph in time exponential in the number of starter dependencies, simply by generating each of the 2^k ways to label each of k starter dependencies in the graph with its LB^- or LB^+, and testing each combination for consistency.

A straightforward algorithm to do this would search the space depth-first with backtracking. Each iteration would pick a starter dependency, force it to LB^- or LB^+, and perform PB again. Continue until either a solution is produced or an inconsistency is found, and then backtrack.

Inconsistencies can only occur at a starter dependency, when either (1) the starter was earlier forced to LB^- (i.e., zero) and PB just found positive support for it, or (2) the starter was forced to LB^+ (i.e., a positive value) and the last support for it was just relabeled zero.

Practical efficiency may be greatly enhanced if the starter dependency is always chosen from a minimal strongly connected component of the unlabeled part of the graph.

Below we consider more sophisticated methods for searching this space.

6.3.3 Consistent and Preferred Extensions

The discussion and algorithm given above indicate that in a stable graph the problem of deciding upon how to resolve the ambiguous nonmonotonic wires is a boolean decision. Thus we should be able to formulate this problem in propositional logic, the satisfying assignments of which would represent the various consistent extensions of the PRIMO specification.

We now present an alternate algorithm, based on propositional satisfiability, for finding consistent extensions. We also show how this algorithm can be used to find an *optimal* extension.

In general, a set of formulae will have many extensions. Given such a set of extensions, some may be preferable to others based on the cost associated with choosing truth values for certain nodes. That is, the LB of the ambiguous antecedents will be coerced to either LB^- or LB^+. We will prefer extensions in which the sum of the differences between their current values to their coerced values is minimized. More formally, let $\neg\boxed{\alpha_i}p_i$ be the set of nonmonotonic premises from a PRIMO rule graph which are still ambiguous after the numeric bounds have been propagated; let $IC(p_i) = \frac{1+LB^+(p_i)-LB^+(\neg p_i)}{2}$.[5] $IC(p_i)$ is a measure of the current approximation of the information content in p_i. An optimal admissible labeling is an admissible labeling that minimizes the objective function:

$$\sum_i |IC(p_i) - LB(p_i)|$$

$LB(p_i)$, the final certainty associated with p_i, will have the value of $LB^+(p_i)$ or $LB^-(p_i)$. Thus the objective function is a measure of the distance of our current numerical approximation to the final value. We want to minimize this distance.

Once we have made the commitment to coercing ambiguous values to either 0 or 1, solving the problem of finding extensions reduces to propositional satisfiability. Extending the problem we consider to that of *weighted satisfiability*, gives us a means of finding a preferred extension. Weighted satisfiability is defined formally below:

[5] This term is equivalent to $\frac{LB^+(p_i)+UB^+(p_i)}{2}$

Let C be a weighted CNF formula, $\bigwedge_i C_i$, where each clause, $C_i = \bigvee_j p_j$, has a corresponding positive weight, w_i. Let P be a truth assignment of the propositional variables, p_i, that appear in C. The weight of P is the sum of the weights of the clauses that are made false by P. The weighted satisfiability problem is to find the minimum weighted truth assignment.

The optimal admissible labeling problem can be encoded as the weighted satisfiability problem in the following way:

Convert the propositional form of the given PRIMO graph into clausal form. Assign infinite weight to each of the resulting clauses. Next, for each ambiguous nonmonotonic premise of the form $\neg \boxed{\alpha_i} p_i$, generate two clauses:

1. p_i with weight $IC(p_i) - LB^-(p_i)$
2. $\neg p_i$ with weight $LB^+(p_i) - IC(p_i)$.

The first clause represents the cost of making p_i false, the second the cost of making p_i true. It is easy to see that the original graph has an admissible labeling if and only if there is a finitely weighted truth assignment for the corresponding instance of weighted satisfiability, and that the weighted truth assignment corresponds to minimizing the objective function given above.

6.4 Example

In this section we demonstrate the above ideas on a simple example. Consider the following rules:

Bird $\wedge \neg \boxed{.2}$ Hops $\rightarrow^{.8}$ Flies

Emu $\wedge \neg \boxed{.2}$ Flies $\rightarrow^{.9}$ Hops

Flemu $\rightarrow^{1}$ Emu

Emu $\rightarrow^{1}$ Bird

Flemu $\rightarrow^{1}$ Flies

Let's say we are given that LB(Bird) = LB(Emu) = 1, and LB(Flemu) = 0.

Then after running the PB algorithm, we obtain that the interval for Flies is [0, .8], and for Hops is [0, .9]. Converting the above rules and inputs into propositional calculus gives us two admissible labelings, Flies $\wedge \neg$ Hops, or $\neg$ Flies $\wedge$ Hops. The optimal one is the latter, which gives us the final labeling: LB(Flies) = 0, LB(Hops) = .9.

Note that if we had started with LB(Emu) = .8, instead of 1, then the optimal labeling would have been: LB(Flies) = .8, LB(Hops) = 0.

6.5 Algorithms and Heuristics

In Section 6.3.3 we showed how the problem we are concerned with can be posed as one of *weighted satisfiability*. Since this problem is intractable in general, we must make compromises if our system is to perform reasonably on nontrivial instances. The alternatives

we consider include constraining the classes of problems we will allow (Section 6.5.1) or sacrificing optimality of solutions (Section 6.5.2).

6.5.1 Nonserial Dynamic Programming

One of the most interesting possibilities involves restricting our attention to classes of formulae which, while still intractable, have satisfiability algorithms which theoretically take much less than $O(2^n)$ time, where n is the number of propositional variables. In [RH86], Hunt and Ravi describe a method based on *nonserial dynamic programming* and *planar separators* (see [BB72] and [LT80], respectively) which solves the satisfiability problem in $O(2^{\sqrt{n}})$ time for a subclass of propositional clauses that can be mapped in a natural way to planar graphs[6]. In [Fer88] Fernandez-Baca discusses an alternative construction for planar satisfiability and an extension to *weighted satisfiability*. He also presents a similar algorithm for another interesting class of problems, where the graph corresponding to the set of clauses has bounded *bandwidth*. Hunt [Hun89] has shown that similar results hold for a large class of problems which have graphs with *bounded channel width*. Each of these is in some sense a measure of the complexity of the clausal form of the problem. If they are much smaller than the number of variables in the problem, weighted satisfiability can be solved relatively quickly for large instances.

6.5.2 Heuristics

Depending on the size of the graph and the deadline imposed on the system by the outside world, we might not afford the time to find the optimal extension. Under these circumstances, we need to use a heuristic that, without guaranteeing an optimal solution, will find a satisficing solution while exhibiting reasonable performance characteristics.[7]

The following heuristics can be applied to the PRIMO graph, after the propagation of bounds, or to the problem encoded in terms of weighted satisfiability.

As initial conditions we assume a set of nodes **P**, which is a subset of the original set of nodes in the graph. Each element of **P** has an associated pair of lower and upper bounds. We sort the elements in **P** such that $\mid IC(p_i) - 0.5 \mid \geq \mid IC(p_{i+1}) - 0.5 \mid$. By sorting the elements in **P** based on decreasing information content, we are trying to first coerce the labeling of those nodes for which we have the strongest constraints.

We can now use a variety of search strategies, such as the iteratively deepening hill-climbing search, or beam-search to (locally) minimize the objective function defined in Section 6.3.3, subjected to the consistency constraints dictated by the graph topology.

[6] It is shown in [Lic82] that the satisfiability problem for this class is NP-complete [GJ79]. Thus the existence of a polynomial time decision procedure is highly unlikely.

[7] As any other heuristic, there is no guarantee that its worst case performance can improve that of an exhaustive search.

6.5.3 Strongly Connected Components

Thus far we have presented our algorithms as if they were to work on the entire PRIMO rule graph. Even the heuristic presented would bog down on rule graphs of realistic size.

As a result, several optimizations are essential in practice, even though they do not affect the theoretical worst case complexity. The entire initial graph can be decomposed into strongly connected components (SCCs), which are attacked one at a time (using whatever algorithm or heuristic is deemed apropriate) "bottom up".

This idea was first used for JTMSs in [Goo87]. As in the JTMS, there is no guarantee that one can avoid backtracking: a low level SCC may have several solutions, and a higher SCC dependent upon it may become unsolvable if the wrong choice is made lower down. However, this strategy seems likely to be helpful in practice.

6.6 Conclusions

We have presented an approach that integrates nonmonotonic reasoning with the use of quantitative information as a criterion for model preference. This represents a major departure from exisiting paradigms, which normally fail to account for one or the other. We have also identified several methods for coping with the inherent intractability involved in such reasoning. We feel that this is a promising approach, but this work is at a preliminary stage. As a result, there are a number of questions which we are considering now. We list some of them below.

- We have previously noted that there are some correspondences between the PRIMO rule graph and that of the JTMS. Their exact relationship (if indeed one exists) is not well understood and needs to be explored.

- The dynamic programming algorithms discussed in Section 6.5.1 may help us to deal with large problem instances under certain structural constraints on the allowed propositional formulae. The results discussed, however, are based on asymptotic bounds. We have begun to implement these algorithms, but do not know at this point whether they will perform satisfactorily in practice. We also need to determine how well the heuristics we have described will perform.

- It may be advantageous to preprocess the graph prior to run time. For instance, breaking up the graph into SCCs may also allow us to do some precomputation at compile time. In addition to generating the SCCs, it might be possible to transform them into canonical forms which would yield more efficient run-time algorithms.

Bibliography

[BB72] Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press, New York, first edition, 1972.

[BB85] Piero P. Bonissone and Allen L. Brown. Expanding the Horizons of Expert Systems. In Thomas Bernold, editor, *Expert Systems and Knowledge Engineering*, pages 267–288, North-Holland, Amsterdam, 1986.

[BBS88] Howard A. Blair, Jr. Allen L. Brown, and V.S. Subrahmanian. *A Logic Programming Semantics Scheme, Part I*. Technical Report LPRG-TR-88-8, Syracuse University School of Computer and Information Science, 1988.

[BD86] Piero P. Bonissone and Keith S. Decker. Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity. In L. Kanal and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 217–247, North-Holland, Amsterdam, 1986.

[BGB87] Allen L. Brown Jr., Dale E. Gaucas, and Dan Benanav. An Algebraic Foundation for Truth maintenance. In *Proceedings 10th International Joint Conference on Artificial Intelligence*, pages 973–980, AAAI, August 1987.

[BW88] Piero P. Bonissone and Nancy C. Wood. Plausible Reasoning in Dynamic Classification Problems. In *Proceedings 1988 Validation and Testing of Knowledge-Based Systems Workshop*, AAAI, August 1988.

[Bon87a] Piero P. Bonissone. Plausible Reasoning: Coping with Uncertainty in Expert Systems. In Stuart Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 854–863, John Wiley and Sons Co., New York, 1987.

[Bon87b] Piero P. Bonissone. Summarizing and Propagating Uncertain Information with Triangular Norms. *International Journal of Approximate Reasoning*, 1(1):71–101, January 1987.

[Dam88] Bruce D'Ambrosio. A Hybrid Approach to Reasoning Under Uncertainty. *International Journal of Approximate Reasoning*, 2(1):29–45, January 1988.

[Doy79] J. Doyle. A truth maintenance system. *Journal of Artificial Intelligence*, 12:2313–272, 1979.

[Eth88] David W. Etherington. *Reasoning with Incomplete Information.* Morgan Kaufmann, San Mateo, CA, first edition, 1988.

[Fer88] David Fernandez-Baca. Nonserial Dynamic Programming Formulations of Satisfiability. *Information Processing Letters*, 27:323–326, 1988.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman, New York, 1979.

[Goo87] James W. Goodwin. *A Theory and System for Non-monotonic Reasoning.* PhD thesis, Linkoping University, 1987.

[Goo88] James W. Goodwin. RUM-inations. 1988. GE-AIP Working Paper.

[Hun89] H. B. Hunt III. personal communication.

[Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[LT80] R. Lipton and R.E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.

[McC86] John McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge. *Journal of Artificial Intelligence*, 28:89–116, 1986.

[Pea88] Judea Pearl. Evidential Reasoning Under Uncertainty. In Howard E. Shrobe, editor, *Exploring Artificial Intelligence*, pages 381–418, Morgan Kaufmann, San Mateo, CA, 1988.

[Rei88] Raymond Reiter. Nonmonotonic Reasoning. In Howard E. Shrobe, editor, *Exploring Artificial Intelligence*, pages 419–481, Morgan Kaufmann, San Mateo, CA, 1988.

[RH86] S.S. Ravi and H.B. Hunt III. *Applications of a Planar Separator Theorem to Counting Problems.* Technical Report 86-19, Suny at Albany, Computer Science Dept., August 1986.

[Sho86] Yoav Shoham. Chronological Ignorance: Time, Nonmonotonicity, Necessity and Causal Theories. In *Proceedings Fifth National Conference on Artificial Intelligence*, pages 389–393, AAAI, August 1986.

7. The Complexity of Horn Theories with Normal Unary Defaults

Jonathan Stillman
Artificial Intelligence Program
General Electric Research and Development Center
P.O. Box 8
Schenectady, N.Y. 12301
e-mail: stillman@crd.ge.com

Abstract

We solve an open problem stated in [KS89], showing that although fast algorithms exist for determining whether a literal holds in a propositional default theory in which the propositional theory consists solely of literals and the default rules are *Horn* (see [KS89]), and exist for deciding satisfiability of propositional Horn theories, the two cannot be combined without introducing intractability. In particular, we show that when the propositional theory of a default theory allows Horn clauses, the membership problem becomes intractable even when the default rules in the theory are restricted to being propositional *normal unary* default rules, a strong restriction of propositional Horn default rules.

We also present several related results, showing that the entailment problem, the enumeration problem, and the problem of determining whether there exists an extension that "satisfies" some specified number of the default rules are all intractable for these restricted default theories.

7.1 Introduction

One of the central concerns of artificial intelligence research involves developing useful models of how one might emulate on computers the 'common-sense' reasoning in the presence of incomplete information that people do as a matter of course. Traditional predicate logics, developed for reasoning about mathematics, are inadequate as a formal framework for such research in that they are inherently *monotonic*: if one can derive a conclusion from a set of formulae then that same conclusion can also be derived from every superset of those formulae. It is argued that people simply don't reason this way: we are constantly making assumptions about the world and revising those assumptions as we obtain more information (see [McC77] or [Min75], for instance).

Many researchers have proposed modifications of traditional logic to model the ability to revise conclusions in the presence of additional information (see, for instance, [McC86], [Moo83], [Poo86]). Such logics are called *nonmonotonic*. Informally, the common idea

in all these approaches is that one may want to be able to "jump to conclusions" that might have to be retracted later. While a detailed discussion of nonmonotonic logics is outside the scope of this paper, a good introduction to the topic can be found in [Eth88], and a number of the most important papers i the field have been collected in [Gin87].

One of the most prominent of the formal approaches to nonmonotonic reasoning, developed by Reiter ([Rei80]), is based on *default rules*, which are used to model decisions made in prototypical situations when specific or complete information is lacking. Reiter's *default logic* is an extension of first order logic that allows the specification of default rules, which we will summarize shortly. Unfortunately, the decision problem for Reiter's default logic is highly intractable in that it relies heavily on consistency checking for processing default rules, and is thus not even semi-decidable (this is not a weakness of Reiter's logic alone; it is common to most nonmonotonic logics). This precludes the practical use of Reiter's default logic in most situations.

The motivation for searching for computationally tractable inference mechanisms for subclasses of *propositional* default reasoning is based on the need to reason about relatively large propositional knowledge bases in which the default structures may be quite simple. Recent research involving inheritance networks with exceptions is particularly relevant, and is explored in depth in [Tou86] and in Chapter 4 of [Eth88], where the close relationship between default logic and inheritance networks with exceptions is explored.

In order to gain computational tractability of reasoning in default logic, one must restrict expressiveness considerably. If one simply restricts the logic to reasoning about arbitrary propositions, the resulting decision problems are at least as hard as deciding standard propositional logic, regardless of any restrictions on the types of default rules allowed. Since the satisfiability problem is intractable for propositional logic, one must consider further restrictions.

Recently, Kautz and Selman ([KS89]) investigated a number of restricted default logics defined over subsets of propositional calculus with various restrictions on the syntactic form of default rules allowed. They described a partial order of such restrictions, and analyzed the complexity of several problems over this partial order when the propositional theory is restricted to a set of literals. Several restrictions on the syntactic form of default rules were shown to result in polynomial-time tests for determining whether certain properties hold given such a restricted propositional theory. In particular, it was shown that one can decide in polynomial time whether there exists an *extension* that contains a given literal when the default rules are restricted to a class they called *Horn* default rules. They suggested that the ability to combine such default theories with non-default propositional Horn theories would be particularly useful, but left open the question of whether the membership problem (i.e., determining whether there exists an extension of a given default theory containing a specified literal) for such a combination of theories is tractable. One of the main theorems of this paper shows that a strong restriction of this problem is NP-complete.

The remainder of this paper is organized as follows: we begin with a brief description of Reiter's default logic, followed by a short overview of NP-completeness, and a

presentation of the restrictions considered by Kautz and Selman. In Section 7.3 we prove that it is NP-complete to determine whether a default theory consisting of non-default propositional Horn clauses together with normal unary default rules contains a given literal. In Section 7.4, we discuss several related results. Finally, we summarize the results presented and discuss areas for further research.

7.2 Preliminaries

7.2.1 Reiter's Default Logic

For a detailed discussion of Reiter's default logic the interested reader is referred to [Rei80]. In this section we will simply review some of the immediately pertinent ideas.

A *default theory* is a pair (D, W), where W is a set of closed well-formed formulae (wffs) in a first order language and D is a set of *default rules*. A *default rule* consists of a triple $< \alpha, \beta, \gamma >$, where

α is a formula called the *prerequisite*,

β is a set of formulae called the *justifications*, and

γ is a formula called the *conclusion*.

Informally, a default rule denotes the statement "if the *prerequisite* is true, and the *justifications* are consistent with what is believed, then one may infer the *conclusion*."

Default rules are written

$$\frac{\alpha : \beta}{\gamma}$$

If the conclusion of a default rule occurs in the justifications, the default rule is said to be *semi-normal*; if the conclusion is identical to the justifications the rule is said to be *normal*.

A default rule is *closed* if it does not have any free occurrences of variables, and a default theory is *closed* if all of its rules are closed.

The maximally consistent sets that can follow from a default theory are called *extensions*. An extension can be thought of informally as one way of "filling in the gaps about the world."

Formally, an extension E of a closed set of wffs T is defined as the fixpoint of an operator , where (T) is the smallest set satisfying:

$W \subseteq (T)$,

(T) is deductively closed,

for each default rule $d \in D$, if the *prerequisite* is in (T), and T does *not* contain the negations of any of the *justifications*, then the *conclusion* is in (T).

Since the operator is not necessarily monotonic, a default theory may not have any extensions. Normal default theories do not suffer from this, however (see [Rei80]), and always have at least one extension.

There are several important properties that may hold for a default theory. Given a default theory (D, W), perhaps together with a literal q, one might want to determine the following about its extensions:

Existence Does there exist *any* extension of (D, W)?

Membership Does there exist an extension of (D, W) that contains q? (This is called *goal-directed reasoning* by Kautz and Selman.)

Entailment Does every extension of (D, W) contain q? (This is closely related to *skeptical reasoning*, where a literal is believed if and only if it is included in all extensions.)

7.2.2 NP-complete Problems

NP is defined to be the class of languages accepted by a nondeterministic Turing machine in time polynomial in the size of the input string. An important subset of NP is the class P, the class of languages accepted by a *deterministic* Turing machine in polynomial time. These problems* comprise those we usually consider *tractable*, in that the time needed to solve them is polynomially related to the problem size.

The "hardest" languages in NP are called NP-complete: NP-complete languages share the property that all languages in NP can be transformed into them via some polynomial time transformation. To show that a problem in NP is NP-complete one must demonstrate a polynomial-time transformation of an instance of a known NP-complete problem to an instance of the problem under consideration in such a way that a solution to one indicates a solution to the other. The known NP-complete problem we will use in this paper is called 3SAT, and is stated formally as follows:

3-SATISFIABILITY (3SAT)

Instance: A finite set $C = \{c_1, \ldots, c_m\}$ of propositional clauses, each of which consists of exactly 3 *literals* (propositional variables or their negations).

Question: Does there exist a truth assignment that satisfies C?

The theory of NP-completeness is relatively well-understood; for a thorough and readable discussion of the topic the interested reader is referred to [GJ79]. The fastest known deterministic algorithms for NP-complete problems take time exponential in the problem size. It is not known whether this is necessary: one of the central open problems in computer science is whether P = NP. Most researchers believe that P $\neq$ NP, and that

*NP-completeness is often discussed in terms of *decision problems* rather than languages, although the two are interchangeable.

NP-complete problems really do need exponential time to solve. Thus these problems are considered *intractable*, since if $P \neq NP$, we cannot hope to solve instances of them with inputs of nontrivial size.

Demonstrating the NP-completeness of a problem does not necessarily imply that it it cannot be solved in practice: sometimes (e.g., the Traveling Salesman Problem) good polynomial approximation algorithms have been devised. Unfortunately, it is not clear what might comprise a reasonable *approximation* to an extension in a default theory. Even when approximation algorithms do not apply, there are often important subclasses of hard problems that can be solved efficiently (deciding satisfiability of propositional Horn clauses is a good example of such a situation). Alternatively, perhaps many of the instances that may arise in practice will have structural properties that can be used to gain tractability. Knowing that a problem is NP-complete is important, however, in that it suggests that exact solutions are unlikely to be obtainable for nontrivial instances, and that some additional restrictions may need to be made on the structure of the problem being considered.

7.2.3 A Taxonomy of Default Theories

In [KS89], Kautz and Selman presented a taxonomy of propositional default theories. They restricted W to contain only propositional literals (i.e., propositional variables and their negations), and restricted default rules to be semi-normal rules in which the precondition, justifications, and conclusions of each default rule consisted of conjunctions of literals (this restriction makes consistency checking a simple task). They also considered the following further restrictions on the default rules allowed.

Unary The prerequisite of each default rule must be a positive literal, and the conclusion must be a literal. If the consequence is positive, the justification must be the conjunction of the consequence and a single negative literal; otherwise, the justification must be the consequence.

Disjunction-Free Ordered The reader is referred to [Eth88] for a formal definition of ordered theories; intuitively, in ordered theories the literals can be ordered in such a way that potentially unresolvable circular dependencies cannot occur.

Ordered Unary These combine the restrictions of the first two theories described above. Kautz and Selman remark that these theories appear to be the simplest necessary to represent inheritance hierarchies with exceptions ([Tou86]).

Disjunction-Free Normal These are disjunction-free ordered theories in which the consequence of each default rule is identical to the justification.

Horn The prerequisite literals in these default rules must each be positive, and the justification and consequence are identical, each consisting of a single literal.

Normal Unary The prerequisite in each of these default rules consists of a single positive literal, the conclusion must be a literal, and the justification must be identical to the consequence. These form the most simple class of default rule that is considered in [KS89].

These restricted theories are related in a partial order as shown in Figure 7.1 below.

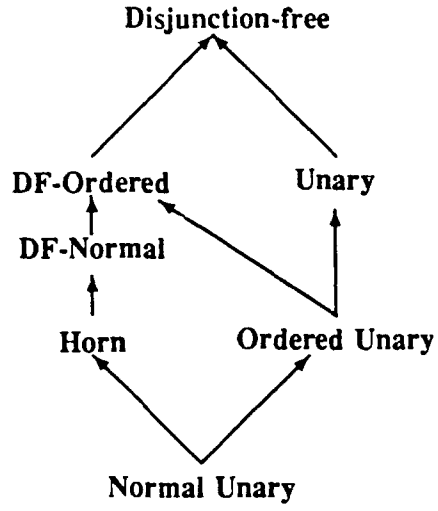


Figure 7.1: Kautz and Selman's hierarchy of restricted default theories.

7.3 Main Results

Quite often, a default theory will have multiple extensions, and one may want to restrict examination to a limited number of them. One important measure of which extensions to consider may be the inclusion of some particular propositions. As mentioned above, this is variously referred to as *goal-directed reasoning* and the *membership problem*. Figure 7.2 summarizes Kautz and Selman's results with regard to the taxonomy they described. In particular, it is shown that for the class of Horn default theories, goal-directed reasoning can be done in linear time when the propositional theory consists of propositional literals. They suggest that although this is somewhat useful, it would be much more interesting if one could combine such default rules with propositional Horn theories efficiently. More formally, one would like to solve the following problem:

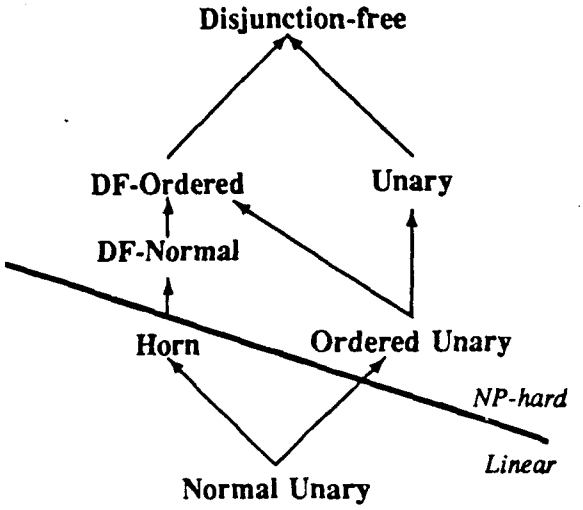


Figure 7.2: The complexity of goal-directed reasoning in the restricted default theories considered by Kautz and Selman.

HORN CLAUSES WITH NORMAL UNARY DEFAULTS

Instance: A *finite set H of propositional* Horn clauses, together with a finite set D of normal, unary, propositional default rules, and a distinguished literal q.

Question: Does there exist an extension of (D, H) that contains the literal q ?

In this section we show that this problem is intractable, proving:

Theorem 1 HORN CLAUSES WITH NORMAL UNARY DEFAULTS *is NP-complete.*

Proof: It is not difficult to demonstrate membership in NP: although the extension may be too large to describe explicitly, it suffices to provide the original set of Horn clauses, together with those default rules that were applied, and verify that the default rules form a maximal set that can actually be applied consistently. Since these are disjunction-free, this can be done efficiently.

To demonstrate NP-hardness we transform an instance of 3SAT to one of HORN CLAUSES WITH NORMAL UNARY DEFAULTS as follows. Given an instance I of 3SAT, we begin by converting I into a new set of clauses consisting of a set H of Horn clauses together with a set P of clauses each of which contain exactly two literals, each occurring positively. To do this we simply place each clause in I that contains at most one positive literal into H; the remaining clauses contain either two or three positive literals. For each of the remaining clauses, choose one of the positive literals (call it x), introduce a new variable $\hat{x}$ and the clauses

$$(\neg x \vee \neg \hat{x}),$$

which is a Horn clause and is placed into H, and

$$(x \vee \hat{x}),$$

which is placed into P. These two clauses taken together are the clausal form of the formula

$$(x \Leftrightarrow \neg\hat{x}).$$

Finally, replace the occurrence of x in the original clause with $\neg\hat{x}$. The resulting clause has one less positive literal than the original; if it is now a Horn clause, place it in H. Otherwise repeat the replacement process to remove one of the remaining positive literals. Note that since equivalence of each literal x and the new corresponding literal $\neg\hat{x}$ is enforced by the added clauses, every satisfying assignment for the original formula can be extended easily to a satisfying assignment for the new formula, and vice versa. The transformation has the property, however, that there are more falsifying assignments for the new formula than for the original. Note also that this transformation only results in a linear increase in the size of the problem.

At this point, we have a set H of Horn clauses, which, together with one more clause we will add later, will comprise the propositional part of the default theory we are constructing. Since the clauses in P are non-Horn, they cannot be included in the propositional part of the theory. Thus, we must construct a set of normal unary default rules D to model the clauses in P. This is done as follows.

For each variable a that appears in some clause in P, we introduce the default rule

$$\frac{:a}{a}$$

into D. Let us assume that P contains m clauses, i.e., $P = \{c_1, \ldots, c_m\}$. Each of these is of the form $c_i = (a \vee b)$, where a and b are positive literals. For each such clause, introduce a new propositional variable q_i, and introduce the following default rules into D:

$$i_1.\ \frac{a : q_i}{q_i} \qquad i_2.\ \frac{b : q_i}{q_i} \qquad i_3.\ \frac{: \neg q_i}{\neg q_i}$$

Once this is done for each of the clauses in P, we introduce one additional new variable and a final Horn clause into H to complete the construction:

$$H_q = (\neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m \vee q)$$

This phase of the transformation also results in at most linear growth in problem size. We now show that there exists an extension of (D, H) that contains q if and only if the original formula F is satisfiable.

($\Rightarrow$). Suppose F is satisfiable. Since we replaced the clauses in P with a set of default rules, we must show that we can, given a satisfying assignment α for F, construct an

extension of (D, H) that contains q. It is easy to see that α can be extended to an assignment α' in which those new variables introduced in transforming F to the sets H and P are assigned truth values so that all the clauses in $H \cup P$ are satisfied, and in fact, that the assignment of values to the new variables is completely determined by α. We use this assignment as the basis for the extension we will construct. We proceed as follows. Each of the clauses in P must have had one of its variables assigned the value *true* by α'. For each of these clauses $c_i = (a \vee b)$ we observe that if a is assigned the value *true* by α', we can apply the default rules

$$\frac{:a}{a} \qquad \frac{a : q_i}{q_i}$$

We can proceed similarly if b is assigned the value *true*. Note that since there are no propositional constraints on the variables q_i other than the single Horn clause we added, we can always consistently add these. When this has been done for each of the clauses, it follows from the Horn clause H_q that the set we have specified also contains q. It is a straightforward matter to confirm that this set can be augmented via deductive closure to form an extension of (D, H) that includes q, since no other default rules can be applied, and the only new Horn clause added (H_q) is also satisfied.

($\Leftarrow$). Suppose that there exists an extension of (D, H) that includes q. Since H contains only non-unit Horn clauses, it is easily seen to be consistent, thus it has only consistent extensions (see [Rei80]). Thus we need only show that each formula from P can be satisfied consistently with H. Since we are given that q is contained in the extension, we can infer from the clause H_q that each of the $\{q_i | 1 \leq i \leq m\}$ are also in the extension (otherwise the extension would contain q_i for some $1 \leq i \leq m$, and the clause H_q would be satisfied without forcing q to be true). For an arbitrary clause $c_i = (a \vee b)$ from P, the default rules

$$\frac{a : q_i}{q_i} \qquad \frac{b : q_i}{q_i}$$

are the only default rules that could have admitted q_i into the extension. The prerequisites of these default rules ensure that at least one of a, b is also in the extension (they may have been included using the default rules

$$\frac{:a}{a} \qquad \frac{:b}{b}$$

or as a consequence of including other literals). Thus, for each of the clauses in P at least one of its literals is in the extension. Since this extension is consistent with H, the set $P \cup H$ is also consistent, and the theorem follows. □

7.4 Related Results

In this section we present several results that can be obtained by making minor modifications to the proof presented above.

Theorem 2 *It is co-NP-complete to determine whether a specified literal q holds in every extension of a default theory (D, H), where H is a finite set of propositional Horn clauses, and D is a finite set of normal, unary, propositional default rules.*

Proof (sketch): The transformation above is modified by adding a default rule

$$\frac{: \neg q}{\neg q}$$

to D, causing the literal $\neg q$ to be included explicitly in every extension if and only if the original instance of 3SAT is unsatisfiable, and the result follows. □

Theorem 3 *It is #-P-complete to count those extensions of a default theory (D, H) containing a specified literal q, where H is a finite set of propositional Horn clauses, and D is a finite set of normal, unary, propositional default rules.*

Proof (sketch): We modify the original transformation by adding default rules corresponding to *each* of the original variables and their negations, rather than just those in P, thus eliminating "don't care" situations that might otherwise arise in extensions, in which for some propositional variables neither the variable or its negation are in the extension. This modified transformation induces a situation in which each extension containing the specified literal q corresponds to a unique satisfying truth assignment for the original formula, and vice versa. The result follows immediately. □

The problems addressed in Theorem 2 and Theorem 3 are closely related to *skeptical reasoning* discussed in [Tou86]. A skeptical reasoning system accepts a proposition only if it is included in *every* extension. It was shown in [KS89] that normal unary default theories have an $O(n^3)$ algorithm for determining whether a proposition holds in all extensions. Theorem 2 demonstrates that if one extends the theory to allow Horn clauses in the non-default part, such skeptical reasoning becomes intractable. Theorem 3 shows that for such default theories it is also intractable to determine whether a proposition holds in *most* extensions. As a result, even approaches approximating skeptical reasoning by accepting propositions that are included in *most* extensions are intractable in these theories.

It is also interesting to note that the construction we describe has the property that for each clause appearing in P, exactly one of the literals in that clause will be true in a given satisfying assignment. The next theorem shows that even determining whether there is an extension that "satisfies" a given number of one's default rules is NP-complete. Since

default rules are often used to express descriptions of "preferred interpretations," such queries provide an indication of how close one might be able to get to one's preferences.

Theorem 4 *It is NP-complete to determine, given a default theory* (D, H)*, where* H *is a finite set of propositional Horn clauses, and* D *is a finite set of normal, unary, propositional default rules with empty prerequisites and with positive justifications and conclusions*[†] *and a positive number* k*, whether there is an extension of* (D, H) *that contains the consequences of at least* k *of the default rules in* D*.*

Proof: The construction of H and P is exactly as in the proof of Theorem 1 above. Note that for each clause $(a \vee b)$ in P there is a corresponding clause $(\neg a \vee \neg b)$ in H. This forces *exactly* one of a and b to be true in any satisfying assignment for $H \cup P$. In order to make sure that applying a default rule corresponds to satisfying *exactly* one clause from P, we must ensure that no variable appears in more than one clause in P. To do this, we proceed as follows. If a variable a appears in two clauses in P, introduce a new variable a', Horn clauses

$$(\neg a \vee a')$$

and

$$(\neg a' \vee a),$$

and replace one occurrence of a in P by a'. When this process is completed, each variable appearing in P appears exactly once in P. Next, for each literal a appearing in P add the default rule

$$\frac{: a}{a}$$

Let m be the number of clauses in P. If the original formula is satisfiable then we can easily extend this to an extension in which exactly m of the default rules were applied, since exactly one of the variables in each clause from P can be true. Similarly, since it is inconsistent for an extension to contain both variables from any clause in P, if there is an extension in which exactly m default rules were applied, exactly one variable from each clause in P is true. Since the clauses in H are consistent, the entire formula is satisfiable. □

7.5 Discussion

We have shown that several problems associated with restricted propositional default theories are intractable, despite the fact that there exist tractable algorithms for their component parts. These default theories are quite simple, and our results show that unless P = NP, in order to effectively reason in default theories one must live with constraints that are quite limiting, some of which are described in [KS89].

[†]These form the most simple possible type of default rule, expressing the desire to believe some propositional variable whenever it is consistent to do so.

The most promising area for further study involves identifying different restrictions that yield tractable reasoning methods without sacrificing expressibility to the point where only trivial default theories can be reasoned about. We are currently investigating several possibilities, and will present a number of new results related to the problem of reasoning in restricted propositional default theories in a forthcoming paper.

Acknowledgements

The author is indebted to Deepak Kapur, Robert Mattheyses, and to the referees for helpful comments on earlier versions of this work.

Bibliography

[Eth88] David W. Etherington. *Reasoning with Incomplete Information*. Pitman, London, 1988.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W.H. Freeman, 1979.

[Gin87] Matthew L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning*. Morgan Kaufman, Los Altos, CA, 1987.

[KS89] Henry A. Kautz and Bart Selman. Hard problems for simple default logics. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 189–197, Toronto, Ontario, Canada, 1989.

[McC77] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1038–1044. International Joint Committee on Artificial Intelligence, 1977.

[McC86] John McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–166, 1986.

[Min75] Marvin Minsky. A framework for representing knowledge. In Patrick Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.

[Moo83] Robert C. Moore. Semantical considerations on nonmonotonic logic. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 272–279, Karlsruhe, West Germany, 1983. International Joint Committee on Artificial Intelligence.

[Poo86] David L. Poole. Default reasoning and diagnosis as theory formation. Technical Report CS-86-08, Dept. of Computer Science, University of Waterloo, 1986.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Tou86] David S. Touretzky. *The Mathematics of Inheritance Systems*. Pitman, London, 1986.

8. It's Not My Default: The Complexity of Membership Problems in Restricted Propositional Default Logics

Jonathan Stillman
Artificial Intelligence Program
General Electric Research and Development Center
P.O. Box 8, Schenectady, N.Y. 12301
e-mail: stillman@crd.ge.com

Abstract

We introduce a hierarchy of classes of propositional default rules, and characterize the complexity of typical problems in those classes under various assumptions about the underlying propositional theory. This work significantly extends both that presented in [KS89] and in [Sti89].

8.1 Introduction

One of the central concerns of artificial intelligence research involves developing useful models of how one might emulate on computers the 'common-sense' reasoning in the presence of incomplete information that people do as a matter of course. Traditional predicate logics, developed for reasoning about mathematics, are inadequate as a formal framework for such research in that they are inherently *monotonic*: if one can derive a conclusion from a set of formulae then that same conclusion can also be derived from every superset of those formulae. It is argued that people simply don't reason this way: we are constantly making assumptions about the world and revising those assumptions as we obtain more information (see [McC77] or [Min75], for instance). Many researchers have proposed modifications of traditional logic to model the ability to revise conclusions in the presence of additional information (see, for instance, [McC86], [Moo83], [Poo86]). Such logics are called *nonmonotonic*. Informally, the common idea in all these approaches is that one may want to be able to "jump to conclusions" which might have to be retracted later. While a detailed discussion of nonmonotonic logics is outside the scope of this paper, a good introduction to the topic can be found in [Eth88], and a number of the most important papers in the field have been collected in [Gin87].

One of the most prominent of the formal approaches to nonmonotonic reasoning, developed by Reiter ([Rei80]), is based on *default rules*, which are used to model decisions made in prototypical situations when specific or complete information is lacking. Reiter's

default logic is an extension of first order logic which allows the specification of default rules, which we will summarize shortly. Unfortunately, the decision problem for Reiter's default logic is highly intractable in that it relies heavily on consistency checking for processing default rules, and is thus not even semi-decidable (this is not a weakness of Reiter's logic alone; it is common to most nonmonotonic logics). This precludes the practical use of Reiter's default logic in most situations.

The motivation for searching for computationally tractable inference mechanisms for subclasses of *propositional* default reasoning is based on the need to reason about relatively large propositional knowledge bases in which the default structures may be quite simple. Recent research involving inheritance networks with exceptions is particularly relevant, and is explored in depth in [Tou86] and in Chapter 4 of [Eth88], where the close relationship between default logic and inheritance networks with exceptions is explored.

In order to gain computational tractability of reasoning in default logic, one must restrict the expressiveness considerably. If one simply restricts the logic to reasoning about arbitrary propositions, the resulting decision problems are at least as hard as deciding standard propositional logic, regardless of restrictions on the types of default rules allowed. Since the satisfiability problem is intractable for propositional logic, one must consider further restrictions. Recently, Kautz and Selman [KS89] and Stillman [Sti89] have investigated default logics defined over subsets of propositional calculus with various restrictions on the syntactic form of default rules allowed. In [KS89], Kautz and Selman described a partial order of such restrictions, and discussed the complexity of several problems over this partial order when the propositional theory is restricted to a set of literals. Several of these restrictions were shown to result in polynomial-time tests for determining whether certain properties hold given such a restricted propositional theory. In particular, it was shown that one can decide in polynomial time whether there exists an *extension* which contains a given literal when the default rules are restricted to a class they called *Horn* default rules. They suggested that the ability to combine such default theories with non-default propositional Horn theories would be particularly useful, but left open the question of whether the membership problem (i.e., determining whether there exists an extension of a given default theory containing a specified literal) for such a combination of theories is tractable. In [Sti89], we showed that a restriction of this problem is NP-complete, and presented several related results.

The remainder of this paper is organized as follows: we begin with a brief description of Reiter's default logic, followed by a short overview of NP-completeness, and a presentation of the restrictions considered by Kautz and Selman. In Section 8.3 we introduce a hierarchy of classes of propositional default rules which significantly extends that presented in [KS89]. Next, we characterize the complexity of the membership problem for these classes. Finally, we summarize the results presented in this paper, and discuss related results and future work.

8.2 Preliminaries

8.2.1 Reiter's Default Logic

For a detailed discussion of Reiter's default logic the interested reader is referred to [Rei80]. In this section we will simply review some of the immediately pertinent ideas. A *default theory* is a pair (D, W), where W is a set of closed well-formed formulae (wffs) in a first order language and D is a set of *default rules*. A *default rule* consists of a triple $< \alpha, \beta, \gamma >$: α is a formula called the *prerequisite*, β is a set of formulae called the *justifications*, and γ is a formula called the *conclusion*. Informally, a default rule denotes the statement "if the *prerequisite* is true, and the *justifications* are consistent with what is believed, then one may infer the *conclusion*." Default rules are written

$$\frac{\alpha : \beta}{\gamma}$$

If the conclusion of a default rule occurs in the justifications, the default rule is said to be *semi-normal*; if the conclusion is identical to the justifications the rule is said to be *normal*. A default rule is *closed* if it does not have any free occurrences of variables, and a default theory is *closed* if all of its rules are closed.

The maximally consistent sets that can follow from a default theory are called *extensions*. An extension can be thought of informally as one way of "filling in the gaps about the world." Formally, an extension E of a closed set of wffs T is defined as the fixpoint of an operator Γ, where $\Gamma(T)$ is the smallest set satisfying:

- $W \subseteq \Gamma(T)$,
- $\Gamma(T)$ is deductively closed,
- for each default $d \in D$, if the *prerequisite* is in $\Gamma(T)$, and T does *not* contain the negations of any of the *justifications*, then the *conclusion* is in $\Gamma(T)$.

Since the operator Γ is not necessarily monotonic, a default theory may not have any extensions. Normal default theories do not suffer from this, however (see [Rei80]), and always have at least one extension.

There are several important properties that may hold for a default theory. Given a default theory (D, W), perhaps together with a literal q, one might want to determine the following about its extensions:

Existence Does there exist *any* extension of (D, W)?

Membership Does there exist an extension of (D, W) which contains q? (This is called *goal-directed reasoning* by Kautz and Selman.)

Entailment Does every extension of (D, W) contain q? (This is closely related to *skeptical reasoning*, where a literal is believed if and only if it is included in all extensions.)

8.2.2 NP-complete Problems

NP is defined to be the class of languages accepted by a nondeterministic Turing machine in time polynomial in the size of the input string. The "hardest" languages* in NP are called NP-complete: all such languages share the property that all languages in NP can be transformed into them via some polynomial time transformation. To show that a problem in NP is NP-complete one must demonstrate a polynomial-time transformation of an instance of a known NP-complete problem to an instance of the problem under consideration in such a way that a solution to one indicates a solution to the other. For a thorough discussion of the topic the interested reader is referred to [GJ79]. The fastest known deterministic algorithms for NP-complete problems take time exponential in the problem size. It is not known whether this is necessary: one of the central open problems in computer science is whether P = NP. Most researchers believe that P $\neq$ NP, and that NP-complete problems really do need exponential time to solve. Thus these problems are considered *intractable*, since if P $\neq$ NP, we cannot hope to solve instances of them with inputs of nontrivial size.

Demonstrating the NP-completeness of a problem does not necessarily suggest that it cannot be solved in practice: sometimes (e.g., the Traveling Salesman Problem) good polynomial approximation algorithms have been devised; unfortunately, it is not clear what might comprise a reasonable *approximation* to an extension in a default theory. Even when approximation algorithms do not apply, there are often important subclasses of hard problems which can be solved efficiently (deciding satisfiability of propositional Horn clauses is a good example of such a situation). Alternatively, perhaps many of the instances that may arise in practice will have structural properties which can be used to gain tractability. Knowing that a problem is NP-complete is important, however, in that it suggests that exact solutions are unlikely to be obtainable for all nontrivial instances, and that some additional restrictions may need to be made on the structure of the problem being considered.

8.2.3 Restricted Default Theories

If practical reasoning systems are to be developed, one cannot ignore computational complexity. Each of the questions mentioned above is at least as hard as deciding the underlying theory W. Thus, if W consists of arbitrary first-order formulae, none of these questions is even semi-decidable, and a practical system must consider stronger restrictions. If W is restricted to arbitrary propositional formulae, each of the questions require deterministic time proportional to that needed to determine propositional satisfiability (approximately 2^n where n is the number of atoms occurring in W, using the best algorithms currently known). It is unlikely that algorithms that perform significantly better will be developed in the future, under the assumption that $p \neq NP$. Thus, to

*NP-completeness is often discussed in terms of *decision problems* rather than languages, although the two are interchangeable.

guarantee efficient answers to the questions posed above, we must consider even stronger restrictions on W. The propositional theories we will consider are described below.

Propositional literals: W consists of propositional atoms and their negations. In [KS89], Kautz and Selman assume this restriction.

Horn clauses: W consists of a conjunction of propositional clauses, each of which contains at most one positive literal.

2-literal clauses: W consists of a conjunction of propositional clauses, each of which contains at most 2 literals. This restriction is assumed in *network default theories*, described by Etherington in [Eth88].

Each of these restricted propositional theories is known to be decidable in linear time[†], providing us with a good starting point for building simple default theories. Note that while the first restriction forms a subset of each of the others, the second and third are incomparable with respect to the formulae they contain. In subsequent sections we will examine the complexity of reasoning in a number of restricted default theories. We will consider default theories for which W falls into one of the three subclasses of propositional formulae presented above. For each of these, we will consider a number of restrictions on what classes of default rules are allowed. These restrictions are discussed below.

8.2.4 Prior Work on Restricted Default Theories

In [KS89], Kautz and Selman presented a taxonomy of propositional default theories. They restricted W to contain only propositional literals, and restricted default rules to be semi-normal rules in which the precondition, justifications, and conclusions of each default rule consisted of conjunctions of literals (this restriction makes consistency checking a simple task). They also considered the following further restrictions on the default rules allowed.

Unary The prerequisite of each default must be a positive literal, and the conclusion must be a literal. If the consequence is positive, the justification must be the conjunction of the consequence and a single negative literal; otherwise, the justification must be the consequence.

Disjunction-Free Ordered We provide a formal definition of *ordered* default theories below; intuitively, in disjunction-free ordered theories the literals can be ordered in such a way that potentially unresolvable circular dependencies cannot occur.

Ordered Unary These combine the restrictions of the first two theories described above[‡].

[†]The first case is trivial. For the second and third, see [DG84] and [APT90], respectively.

[‡]Kautz and Selman remark that these theories appear to be the simplest necessary to represent inheritance hierarchies with exceptions ([Tou86]).

Disjunction-Free Normal These are disjunction-free ordered theories in which the consequence of each default rule is identical to the justification.

Horn The prerequisite literals in these defaults must each be positive, and the justification and consequence are each a single literal.

Normal Unary The prerequisite in each of these defaults consists of a single positive literal, the conclusion must be a literal, and the justification must be identical to the consequence. These form the most simple class of default rule that is considered in [KS89].

Ordered default theories are discussed in detail in [Eth87]; some of our results relate to such theories, so a definition is provided below. First, we need to define two relations on literals, $\ll$ and $\lesssim$. These are defined on closed, semi-normal default theories $\Delta = (D, W)$, assumed without loss of generality to be presented in clausal form.

1. If $\alpha \in W$ then $\alpha = (\alpha_1 \vee \ldots \vee \alpha_n)$, for some $n \geq 1$. For $\alpha_i \neq \alpha_j$, let $\neg\alpha_i \lesssim \alpha_j$.

2. If $\delta = \dfrac{\alpha : \beta \wedge \gamma}{\beta} \in D$, let $\alpha_1, \ldots, \alpha_r, \beta_1, \ldots, \beta_s, \gamma_1, \ldots, \gamma_t$ be the literals appearing in the clauses of α, β, and γ, respectively. Then

   (i) For $\alpha_i \in \{\alpha_1, \ldots, \alpha_r\}, \beta j \in \{\beta_1, \ldots, \beta_s\}$, let $\alpha_i \lesssim \beta_j$.

   (ii) For $\gamma_i \in \{\gamma_1, \ldots, \gamma_r\}, \beta j \in \{\beta_1, \ldots, \beta_s\}$, let $\neg\gamma_i \ll \beta_j$.

   (iii) $\beta = \beta_1 \wedge \ldots \wedge \beta_m$ for some $m \geq 1$. For each $\beta_i = (\beta_{i,1} \vee \ldots \vee \beta_{i,k_i}) \in \beta$, if $\beta_{i,j} \neq \beta_{i,k}$, let $\neg\beta_{i,j} \lesssim \beta_{i,k}$.

3. The expected transitivity relationships hold for $\ll$ *and* $\lesssim$. Thus,

   (i) If $\alpha \lesssim \beta$ and $\beta \lesssim \gamma$, then $\alpha \lesssim \gamma$.

   (ii) If $\alpha \ll \beta$ and $\beta \ll \gamma$, then $\alpha \ll \gamma$.

   (iii) If $\alpha \ll \beta$ and $\beta \lesssim \gamma$, or $\alpha \lesssim \beta$ and $\beta \ll \gamma$, then $\alpha \ll \gamma$.

A semi-normal default theory $\Delta = (D, W)$ is said to be *ordered* if and only if there is no literal α such that $\alpha \ll \alpha$.

These restricted theories are related in a partial order as shown in Figure 8.1 below. Kautz and Selman examined the extension existence, membership, and entailment questions for these theories.

Prompted by a gap in the characterization of restricted default theories, in our recent paper ([Sti89]) we showed that the following problem is NP-complete.

Horn Clauses with Normal Unary Defaults (HC-NU)
Instance: A finite set H of propositional Horn clauses, together with a finite set D of normal, unary, propositional defaults, and a distinguished literal q.

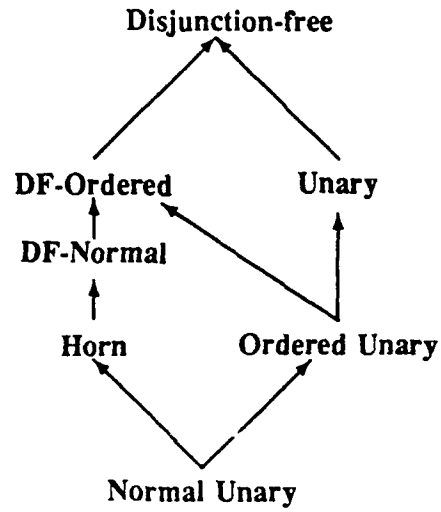


Figure 8.1: Kautz and Selman's hierarchy of default theories.

Question: Does there exist an extension of (D, H) which contains q ?

This result subsumed an open question cited in [KS89]: Kautz and Selman were interested in whether one could add Horn defaults to Horn propositional theories without introducing intractability. Unfortunately, our result answers this question negatively. It was also shown that the entailment problem is co-NP-complete for these default theories.

We subsequently examined even stronger restrictions on the classes of default rules allowed, hoping to find a class of rules which could be combined with Horn clauses while retaining the tractability of propositional Horn clause reasoning. We also examined the complexity of restricted default reasoning under other restrictions on the propositional theories allowed, as described above. In the following sections, we report on the results of this work.

8.3 Expanding the Horizons

Our investigation suggested a richer hierarchy of default rules, most of which result from disallowing any prerequisites in rules. This corresponds to introducing a "context-free" element to the reasoning. In this section, we explore the complexity of membership problems in default theories in which W belongs to one of the classes of formulae listed above, and in which D belongs either to one of the classes of default rules discussed above or to one of the following:

Prerequisite-Free Disjunction-free default rules with no prerequisites.

Prerequisite-Free Unary The prerequisite of each rules is empty, and the conclusion of each default must be a literal. If the consequence is positive, the justification must be the conjunction of the consequence and a single negative literal; otherwise, the justification must be the consequence.

Prerequisite-Free Ordered A prerequisite-free ordered theories is a disjunction-free ordered theory in which the prerequisite is empty.

Prerequisite-Free Ordered Unary These combine the restrictions of the first two theories described above.

Prerequisite-Free Normal These are prerequisite-free ordered theories in which the consequence of each default rule is identical to the justification.

Prerequisite-Free Normal Unary The prerequisite in each of these defaults is empty, the conclusion must be a literal, and the justification must be identical to the consequence.

Prerequisite-Free Positive Normal Unary The prerequisite in each of these defaults is empty, the conclusion must be a positive literal, and the justification must be identical to the consequence.

These restricted theories are related in a partial order. The hierarchy is shown in Figure 8.2 below.

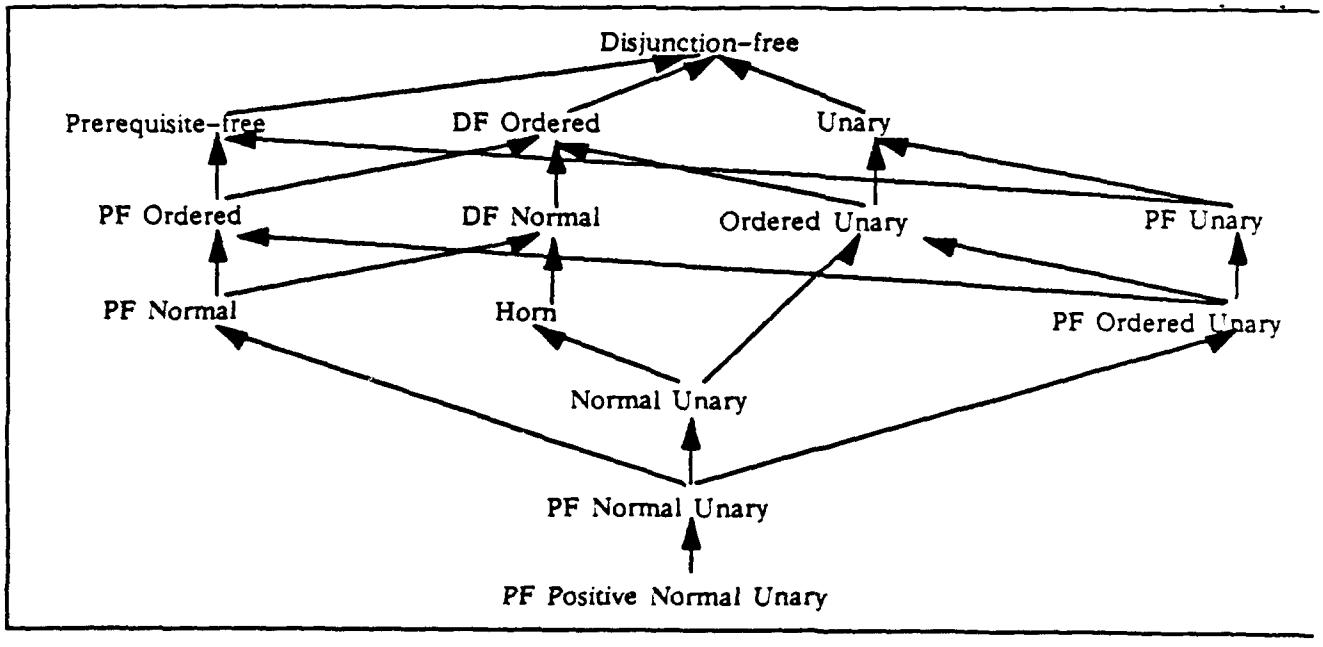


Figure 8.2: An expanded hierarchy of default rules.

8.4 Horn Clause Theories

After showing that the problem **HC-NU** was NP-complete, we looked for even tighter restrictions on the defaults allowed which would provide us with tractable default reasoning where the propositional theory consisted of Horn clauses. Unfortunately, our results here were quite negative. The membership problem remains intractable under very tight restrictions. In particular, for the following problem -

Horn Clauses with Prerequisite-Free Positive Normal Unary Defaults (HC -2)
Instance: A finite set H of propositional Horn clauses, together with a finite set D of prerequisite-free positive normal, unary, propositional defaults, and a distinguished literal q.
Question: Does there exist an extension of (D, H) which contains q ?
we prove:

Theorem 1 *HC-2 is NP-complete.*

Proof: It is not difficult to demonstrate membership in NP: although the extension may be too large to describe explicitly, it suffices to provide the original set of Horn clauses, together with those defaults that were applied, and verify that the defaults can actually be applied consistently. Since these are disjunction-free, this can be done efficiently.

To demonstrate NP-hardness we transform an instance of NOT-ALL-EQUAL SATISFIABILITY to one of HC-2. NOT-ALL-EQUAL SATISFIABILITY can be stated as follows.

> Given sets $S_1, S_2, \ldots, S_m$, each having 3 members, can the members be colored with two colors so that no set is all one color?

In [Sha78] it is shown that NOT-ALL-EQUAL SATISFIABILITY is NP-complete. Given an instance I of NOT-ALL-EQUAL SATISFIABILITY, let Σ be the set of all elements appearing in any S_j. For each such element σ_i, introduce the a new propositional atom σ_i, and add the following default rule to D:

$$\frac{: \sigma_i}{\sigma_i}$$

Next, for each set $S_j = \{s_{j_1}, s_{j_2}, s_{j_3}\}$ in I introduce a new propositional atom S_j, and add the following clauses to W:

$$(\neg s_{j_1} \vee \neg s_{j_2} \vee \neg s_{j_3})$$
$$(\neg s_{j_1} \vee S_j)$$
$$(\neg s_{j_2} \vee S_j)$$
$$(\neg s_{j_3} \vee S_j)$$

Finally, introduce a new propositional atom q and add the following clause to W:

$$(\neg S_1 \vee \neg S_2 \vee \ldots \vee \neg S_m \vee q),$$

This completes the transformation, which results in only a linear increase in the size of the problem.

We now show that there exists an extension of (D, W) which contains q if and only if the original instance I of NOT-ALL-EQUAL SATISFIABILITY is satisfiable.
($\Rightarrow$). Suppose I is satisfiable. Then the elements of Σ must be two-colorable in such a way that none of the sets S_j has all its elements the same color. Let us assume that the two colors correspond to the truth values *true* and *false*. There must exist a satisfying assignment to the elements of Σ in which a maximal number of the elements of Σ are colored *true*. We must show that we can, given such a maximal satisfying assignment α for I, construct an extension of (D, W) which contains q.

We proceed as follows. Each of the sets in S must have had at least one of its elements assigned the value *true*. For each such element, assign the corresponding atom in the instance of H-2 the value *true*. This can be done using the default rules which were added for each of the set elements. It is not hard to see that this can always be done consistently: the three element clauses introduced into W will not be contradicted, since they correspond to at least one of the elements of each set being assigned the value *false*. We know that this is the case since we are given a solution to I. Since the assignment in I is maximal, no other set elements can be made true without forcing at least one of the sets to have all its elements take the same value. Thus, none of the remaining default rules can be applied. Since each set has at least one of its members assigned the value *true*, each of the propositional atoms S_i are true in the extension we are constructing.
($\Leftarrow$). Suppose there exists an extension of (D, W) which contains q. It follows that each of the literals of the form $S_j : 1 \leq j \leq m$ must be *true* (this is the only way to force q to be *true*). Furthermore, it follows that for each such literal, S_j, at least one of the literals in the set $\{s_{j_1}, s_{j_2}, s_{j_3}\}$ must be *true*. The clause in W of the form

$$(\neg s_{j_1} \vee \neg s_{j_2} \vee \neg s_{j_3})$$

forces at least one of these to be *false* as well. This provides us with at least one element of each set $S_j : 1 \leq j \leq m$ which is *true*, and at least one which is *false*. From this it is easy to construct a satisfying assignment for for the instance I of NOT-ALL-EQUAL-SATISFIABILITY. □

The implications of this result on the hierarchy above are summarized in Figure 8.3 below.

8.5 2-Literal Clauses

A second interesting subclass of propositional formulae is 2-literal clauses. The classes formed by combining theories consisting of 2-literal clauses with restricted default theories is assumed in *network default theories*, described by Etherington in [Eth88]. We

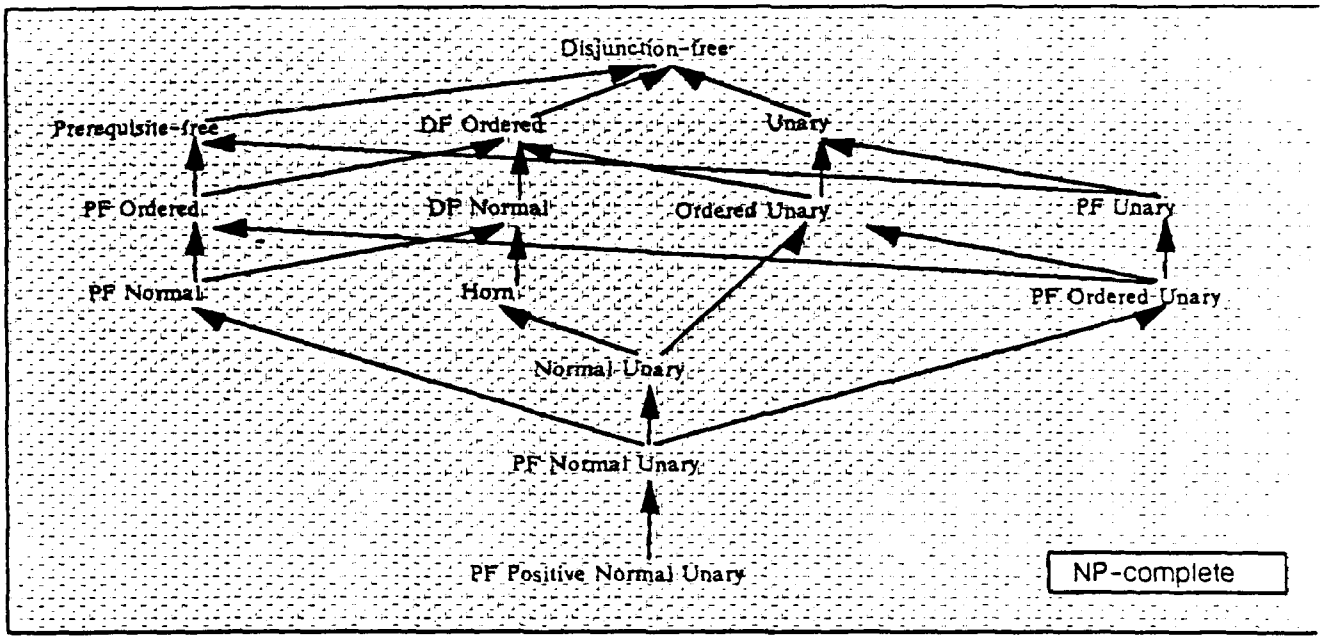


Figure 8.3: The complexity of membership problems with Horn theories.

investigated the complexity of membership problems for this class given the above hierarchy of restrictions on D shown above. For the problem

2-Literal Prerequisite-Free Normal
Instance: A finite set W of propositional 2-literal clauses, together with a finite set D of prerequisite-free normal propositional defaults, and a distinguished literal q.
Question: Does there exist an extension of (D, W) which contains q ?
we have the following theorem:

Theorem 2 *2-Literal Prerequisite-Free Normal can be solved in polynomial time.*

We present an $O(n^3)$ algorithm deciding the membership problem for this class in [Sti90]. The basic idea is to exploit the structural property of 2-literal clauses that they resemble binary relations. As a result, we can effectively compute an implicational "closure" of the underlying propositional theory. Once this is done, it is relatively easy to determine whether there is a default rule which can be used to force q to be included in the extension. For the probelem

2-Literal Normal Unary
Instance: A finite set W of propositional 2-literal clauses, together with a finite set D of normal unary propositional defaults, and a distinguished literal q.
Question: Does there exist an extension of (D, W) which contains q ? we prove the following:

Theorem 3 *2-Literal Normal Unary is NP-complete.*

Proof: As above, membership in NP is fairly straightforward. To demonstrate NP-hardness, we transform an instance I of 3-SAT to the membership problem for 2-Literal Normal Unary defaults. The problem of 3-SAT is stated formally as follows:
3-SATISFIABILITY (3-SAT)

Instance: A finite set $C = \{c_1, \ldots, c_m\}$ of propositional clauses, each of which consists of exactly 3 *literals* (propositional atoms or their negations).

Question: Does there exist a truth assignment that satisfies C?

We proceed as follows. Given an instance I of 3-SAT, let $C = \{C_1, C_2, \ldots, C_m\}$ be the clauses appearing in I, and let V be the set of all propositional atoms appearing in any clause C_j. For each clause $C_j = \{l_{j_1}, l_{j_2}, l_{j_3}\}$ (where each l_{j_k} is either a propositional atom from V or its negation) introduce new atoms $\{c_{j_1}, c_{j_2}, c_{j_3}\}$ together with the clauses

$$(\neg c_{j_1} \vee l_{j_1})$$
$$(\neg c_{j_2} \vee l_{j_2})$$
$$(\neg c_{j_3} \vee l_{j_3})$$

It is important to note that these clauses can never force any of the atoms c_{i_j} to be assigned the value *true*. If the corresponding literal l_{i_j} is true, the clause is satisfied; if it is false, c_{i_j} must be made false to satisfy the clause. We assume that the clauses are ordered according to their subscripts (the order in which they appear). We add default rules to D as follows:
For C_1, add the rules

$$\frac{: c_{1_1}}{c_{1_1}} \qquad \frac{: c_{1_2}}{c_{1_2}} \qquad \frac{: c_{1_3}}{c_{1_3}}$$

Next, for each clause $C_i : 1 < i \leq m$, add the default rules

$$\frac{c_{(i-1)_1} : c_{i_1}}{c_{i_1}} \qquad \frac{c_{(i-1)_1} : c_{i_2}}{c_{i_2}} \qquad \frac{c_{(i-1)_1} : c_{i_3}}{c_{i_3}}$$

$$\frac{c_{(i-1)_2} : c_{i_1}}{c_{i_1}} \qquad \frac{c_{(i-1)_2} : c_{i_2}}{c_{i_2}} \qquad \frac{c_{(i-1)_2} : c_{i_3}}{c_{i_3}}$$

$$\frac{c_{(i-1)_3} : c_{i_1}}{c_{i_1}} \qquad \frac{c_{(i-1)_3} : c_{i_2}}{c_{i_2}} \qquad \frac{c_{(i-1)_3} : c_{i_3}}{c_{i_3}}$$

Finally, introduce a new propositional atom q and add the following default rules to D:

$$\frac{c_{m_1} : q}{q} \qquad \frac{c_{m_2} : q}{q} \qquad \frac{c_{m_3} : q}{q}$$

This completes the transformation, which results in only a linear increase in the size of the original problem. It is easy to see that both D and W satisfy the constraints of the problem statement. We now show that there exists an extension of (D, W) which contains q if and only if the original instance I of 3-SAT is satisfiable.
($\Rightarrow$). Suppose I is satisfiable. Then there is some assignment of truth values to the atoms in I such that at least one literal in each clause is assigned the value *true*. It follows immediately that at least one of each of the clauses added to W for each C_i is satisfied via one of the $l_{i_{k_i}} : (1 \leq k_i \leq 3)$. Note that this leaves us free to assign the corresponding atom $c_{i_{k_i}}$ the value *true*, assuming the default rules can be used to do so. We do this by applying the correct default rule for clause C_1, making $c_{1_{k_1}}$ true. This enables us to use the appropriate default rule for c_2 to make $c_{2_{k_2}}$ true. This process continues until the default rules have been used to make some $c_{i_{k_i}}$ true for $1 \leq i \leq m$. In particular, one of $\{c_{m_1}, c_{m_2}, c_{m_3}\}$ has been made true, which allows us to make q true. Although this may not yet be an extension since some other $c'_{i_j}s$ may still be made true consistently, it is easily seen that any some extension will always result from this process, and that that extension will contain q (one can also simply add the default rules

$$\frac{: \neg c_{1_1}}{\neg c_{1_1}} \qquad \frac{: \neg c_{1_2}}{\neg c_{1_2}} \qquad \frac{: \neg c_{1_3}}{\neg c_{1_3}}$$

which allows us to make two of these three atoms false in each extension).
($\Leftarrow$). Suppose there exists an extension of (D, W) which contains q. Since the only place that q appears is in the default rules with antecedents from C_m, it must be the case that at least one of these has been assigned the value *true* in the extension. As mentioned above, none of the clauses in W can force these atoms to be assigned *true*; thus it must be the case that one of the atoms from $C_{(m-1)}$ has been made true, allowing one of the default rules constructed for $C_{(m-1)}$ to be applied. This reasoning can be continued downward through the default rules for C_1; in this way we can show that for each $C_i : 1 \leq i \leq m$ at least one of the atoms $c_{i_k} : 1 \leq k \leq 3$ is true in the extension. Since each of the clauses in W is satisfied in the extension, one can see that for each c_{i_k} which is true in the extension, there is a corresponding literal l_{i_k} which is also true. We can obtain a satisfying assignment for I by making exactly these literals true in I. □

2-Literal Prerequisite-Free Ordered Unary
Instance: A finite set W of propositional 2-literal clauses, together with a finite set D of prerequisite-free ordered unary propositional defaults, and a distinguished literal q.
Question: Does there exist an extension of (D, W) which contains q ?

Theorem 4 *2-Literal Prerequisite-Free Ordered Unary is NP-complete.*

Proof: Once again, demonstrating membership in NP is fairly straightforward. To demonstrate NP-hardness, we transform an instance I of 3-SAT to the membership problem for 2-Literal Prerequisite-Free Ordered Unary defaults as follows.

Given an instance I of 3-SAT, let $C = \{C_1, C_2, \ldots, C_m\}$ be the clauses appearing in I, and let V be the set of all propositional atoms appearing in any clause C_j. For each clause $C_j = \{l_{j_1}, l_{j_2}, l_{j_3}\}$ (where each l_{j_k} is either a propositional atom from V or its negation) introduce a new atom C_j. We will also need two other new atoms, q and inc. For each clause $C_j : (1 \leq j \leq m)$ we add the following clauses to W:

$$(C_j \vee \neg l_{j_1})$$
$$(C_j \vee \neg l_{j_2})$$
$$(C_j \vee \neg l_{j_3})$$

and add the following default rules to D:

$$\frac{: inc \wedge \neg C_j}{inc}$$

Next, we add the following default rule to D:

$$\frac{: q \wedge \neg inc}{q}$$

Finally, for each literal l_{i_j} that appears in some clause in I, add the default rule:

$$\frac{: l_{i_j}}{l_{i_j}}$$

This completes the transformation. The transformed instance is linearly related to the original. It is easy to see that (D, W) is 2-Literal Prerequisite-Free Unary. It is also easy to see that it is ordered: the strongest (most restrictive) possible relation that can hold between literals appearing in W (and their complements), is that they are all related to one another via $\lesssim$. The only literals that appear in semi-normal default rules are the literals C_i, inc, and q. The default rules force $C_i \ll inc \ll q$. Since neither inc nor q appear in W, they cannot be $\ll$ or $\lesssim$ to any of the other literals. Thus, in even the most restrictive relation possible, we have an ordered theory.

We now show that there exists an extension of (D, W) which contains q if and only if the original instance I of 3-SAT is satisfiable.

($\Rightarrow$). Suppose I is satisfiable. Then there is some assignment of truth values to the atoms in I such that at least one literal in each clause is assigned the value *true*. It follows immediately that at least one of each of the clauses added to W for each C_i must be satisfied making the atom C_i true (since there is a clause in W of the form

$$(C_i \vee \neg l_{i_j})$$

where l_{i_j} is a literal assigned the value *true* in the satisfying assignment for I. As a result, none of the default rules that make the atom inc true can be applied. Thus, since

there are no constraints on q, the default rule

$$\frac{: q \wedge \neg inc}{q}$$

can be applied, making q true in the extension. The extension can be completed by applying the default rules introduced for the literals in I as applicable. This cannot interfere with the inclusion of q.
($\Leftarrow$). Suppose there exists an extension of (D, W) which contains q. Since the only place that q appears is in the default rule

$$\frac{: q \wedge \neg inc}{q}$$

it must be the case that it is consistent to believe $\neg inc$. Thus, it must be the case that none of the default rules of the form

$$\frac{: inc \wedge \neg C_j}{inc}$$

can be applied, so it must be inconsistent to believe the negation of any of the clause atoms C_i. It follows that for each clause, at least one of the literals in that clause is true, made so by applying the appropriate default rule for that literal in creating the extension. We can easily obtain a satisfying assignment for I by making exactly these literals true in I. □

These results are summarized in Figure 8.4 below.

8.6 Single Literal Theories

As mentioned above, this is the class that was investigated in [KS89]. The complexity of reasoning in the theories they considered is described in [KS89]; their results, together with ours, are illustrated in Figure 8.5 below. Since these theories are contained in both of those considered above, problems easy for them are also easy for these. The new result we present for these theories is given below:

Single Literal Prerequisite-Free Ordered Unary
Instance: A finite set W of propositional single literal clauses, together with a finite set D of prerequisite-free ordered unary propositional defaults, and a distinguished literal q.
Question: Does there exist an extension of (D, W) which contains q ?

Theorem 5 *Single Literal Prerequisite-Free Ordered is NP-complete.*

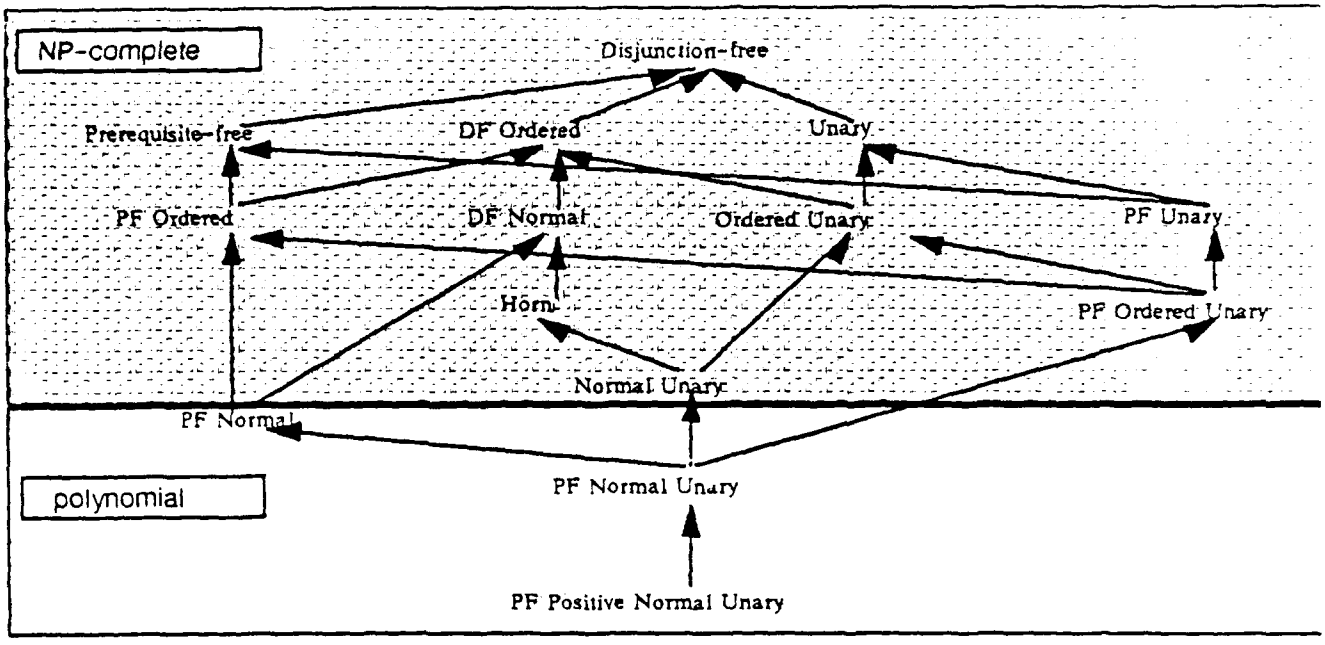


Figure 8.4: The complexity of membership problems with 2-literal theories.

Proof: Once again, demonstrating membership in NP is fairly straightforward. To demonstrate NP-hardness, we transform an instance I of 3-SAT to the membership problem for Single Literal Prerequisite-Free Ordered defaults as follows.

Given an instance I of 3-SAT, let $C = \{C_1, C_2, \ldots, C_m\}$ be the clauses appearing in I, and let V be the set of all propositional atoms appearing in any clause C_j. Using two new atoms, q and inc, we add the following default rule to D:

$$\frac{: q \wedge \neg inc}{q}$$

Next, for each clause $C_i : 1 \leq i \leq m$, introduce a new atom c_i and a default rule:

$$\frac{: inc \wedge \neg c_i}{inc}$$

For each clause $C_i = \{l_{i_1}, l_{i_2}, l_{i_3}\}$ (where each l_{i_j} is either a propositional atom from V or its negation) we add the following default rules to D:

- If l_{i_j} is the negation of a propositional atom a, add the rule

$$\frac{: c_i \wedge \neg a}{c_i}$$

- If l_{i_j} is a propositional atom a, add a new atom a_{i_j} and the rules

$$\frac{: c_i \wedge \neg a_{i_j}}{c_i}$$

and

$$\frac{: a_{i_j} \wedge \neg a}{a_{i_j}}$$

Finally, for each atom a that appears either positively or negatively in some clause in I, add the default rules:

$$\frac{: a}{a} \qquad \frac{: \neg a}{\neg a}$$

This completes the transformation. The transformed instance is linearly related to the original. It is easy to see that (D, W) is Single-Literal Prerequisite-Free Ordered. We show that it is ordered below.

We now show that there exists an extension of (D, W) which contains q if and only if the original instance I of 3-SAT is satisfiable.
($\Rightarrow$). Suppose I is satisfiable. Then there is some assignment of truth values to the atoms in I such that at least one literal in each clause is assigned the value *true*. We proceed by applying the default rule corresponding to each literal which is true in I (it is easy to see that one can always do this consistently). Consider the default rules that were introduced to the transformation for each clause. Each of the original clauses C_i has at least one literal assigned true in the assignment; we consider two cases for each such clause:

1. If there is a negative literal $\neg a$ occurring in C_i which is true in the assignment, there is a corresponding rule of the form

$$\frac{: c_i \wedge \neg a}{c_i}$$

which can be applied (since we applied the default rule corresponding to $\neg a$, and there are no constraints on the atoms c_i which prohibit us from applying this rule).

2. For any remaining clauses C_i, there is only a positive literal a_{i_j} occurring in C_i which is true in the assignment. There are corresponding default rules of the form

$$\frac{: c_i \wedge \neg a_{i_j}}{c_i}$$

and

$$\frac{: a_{i_j} \wedge \neg a}{a_{i_j}}$$

both of which can be applied (since we applied the default rule corresponding to a_{i_j}, and there are no constraints on the atoms c_i which prohibit us from applying these rules).

This leaves us with a partial extension containing each $ci : 1 \leq i \leq m$ together with each of the literals true in the assignment. Since each default rule that allows us to include inc in an extension relies on the consistency of including the negation of one of the atoms c_i, no default rule can be applied to include inc. There are no other constraints on inc, so $\neg inc$ is consistent with the extension we are constructing, as is q. Thus, none of the default rules that make the atom inc true can be applied. Since there are no constraints on q, the default rule

$$\frac{: q \wedge \neg inc}{q}$$

can be applied, making q true. It is now a straightforward matter to check that this results in an extension that contains q.
($\Leftarrow$). Suppose there exists an extension of (D, W) which contains q. Since the only place that q appears is in the default rule

$$\frac{: q \wedge \neg inc}{q}$$

it must be the case that it is consistent to believe $\neg inc$. Thus, it must be the case that none of the default rules of the form

$$\frac{: inc \wedge \neg c_i}{inc}$$

can be applied, so it must be inconsistent to believe the negation of any of the literals for each clause (i.e., each clause atom $c_i : 1 \leq i \leq m$ is in the extension). It follows that for each clause, at least one of the literals in that clause is true, made so by applying the appropriate default rule (or rules, if the literal is positive in the clause) for that literal in creating the extension. We can easily obtain a satisfying assignment for I by making exactly these literals true in I. □

These results are summarized in Figure 8.5 below.

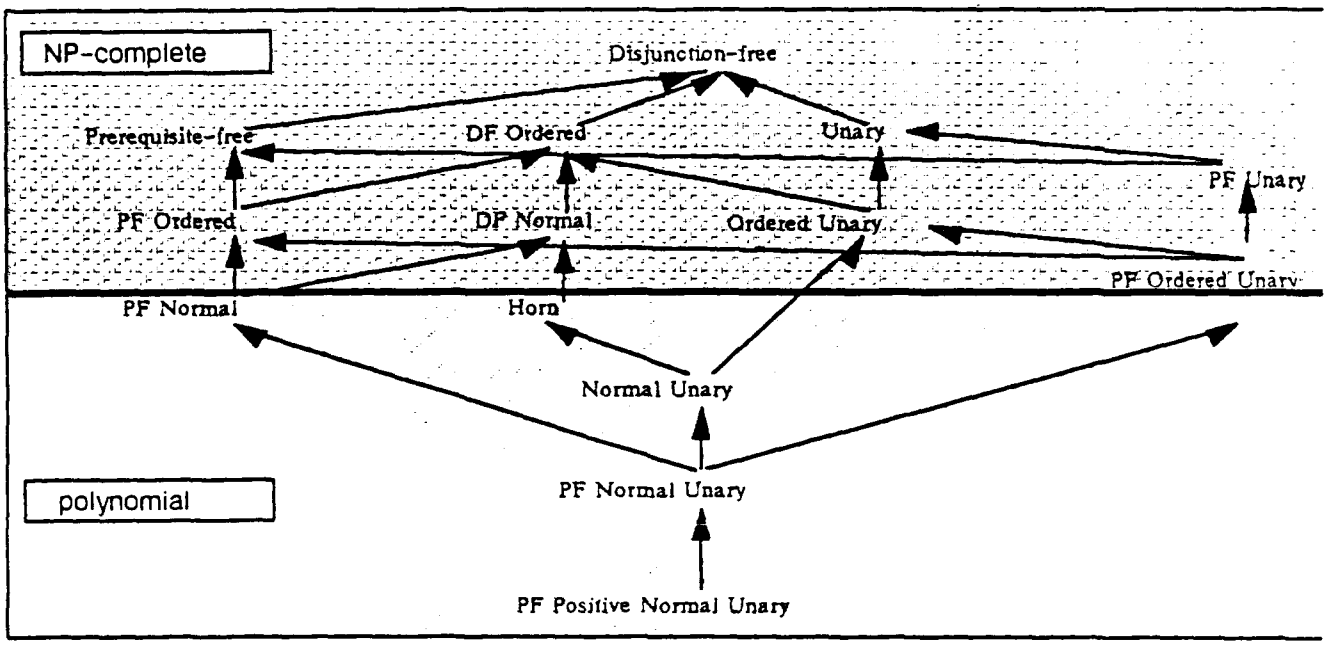


Figure 8.5: The complexity of membership problems with single literal theories.

8.7 Conclusions and Future Research

We have presented a number of results which characterize the complexity of the membership problem for restricted default theories. This work significantly extends that presented in [KS89] and [Sti89]. Our work considers very tight restrictions on the expressiveness of default rules as well as the underlying propositional theory. Unfortunately, our results show that even under these restrictions, membership problems almost invariably remain intractable. This suggests that if practical default reasoning systems are desired, one must either consider extremely restricted expressiveness or work to identify subcases of otherwise intractable classes which yield feasible complexity.

Most of the questions regarding extension existence and entailment over the classes we have considered can be answered as corollaries to the results we have presented. This is addressed in the full version of this paper ([Sti90]). The reader will note, however, that we have left two questions unanswered at this time, those being the complexity of theories consisting of prerequisite-free unary and ordered unary defaults when the underlying propositional theory consists of single literals. We are currently investigating these questions.

Bibliography

[APT90] Aspvall, V., Plass, M.F., and Tarjan, R.E., "A linear-time algorithm for testing the truth of certain quantified Boolean formulas," *Information Processing Letters* 8 (3), 1979.

[DG84] Dowling, W.F., and Gallier, J.H., "Linear time algorithms for testing the satisfiability of propositional horn formulas," *Journal of Logic Programming*, 3:267-284, 1984.

[Eth87] Etherington, D.W., "Formalizing Non-Monotonic Reasoning Systems," *Artificial Intelligence* 31:41-85, 1987.

[Eth88] Etherington, D.W., *Reasoning with Incomplete Information*, Pitman, London, 1988.

[GJ79] Garey, M.R., and Johnson, D.S., *Computers and Intractability*, W.H. Freeman, 1979.

[Gin87] Ginsberg, M.L., (editor), *Readings in Nonmonotonic Reasoning*, Morgan Kaufman, Los Altos, CA, 1987.

[KS89] Kautz, H.A., and Selman, B., "Hard Problems for Simple Default Logics," *Proc. First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ontario, Canada, 1989, pp. 189-197.

[McC77] McCarthy, J., "Epistemological Problems of Artificial Intelligence," *Proc. Fifth International Joint Conference on Artificial Intelligence*, 1977, pp. 1038-1044.

[McC86] McCarthy, J., "Applications of Circumscription to Formalizing Commonsense Knowledge," *Artificial Intelligence 28*, 1986, pp. 89-166.

[Min75] Minsky, M., "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, P. Winston (ed.), McGraw-Hill, New York, 1975, pp. 211-277.

[Moo83] Moore, R., "Semantical considerations on nonmonotonic logic," *Proc. 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, 1983, pp. 272-279.

[Poo86] Poole, D.L., *Default Reasoning and Diagnosis as Theory Formation*, Technical Report CS-86-08, Dept. of Computer Science, University of Waterloo, 1986.

[Rei80] Reiter, R., "A Logic for Default Reasoning," *Artificial Intelligence 13*, North-Holland, 1980, pp. 81-132.

[Sha78] Shaefer, T.J., "The Complexity of Satisfiability Problems," in *Proceedings of the 10th Annual ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, 1978.

[Sti89] Stillman, J.P., *"Horn Theories with Normal Unary Defaults are Intractable,"* to be presented at the 8th Canadian Artificial Intelligence Conference, (CSCSI/SCEIO-90), Ottawa, Canada, 23-25 May 1990 (also available as a General Electric Research and Development Center Technical Report, #89CRD243).

[Sti90] Stillman, J.P., *"The Complexity of Reasoning in Restricted Propositional Default Logics,"* General Electric Research and Development Center Technical Report, in preparation.

[Tou86] Touretzky, D., *The Mathematics of Inheritance Systems*, Pitman, London, 1986.

9. PRIMO: A Tool for Reasoning with Incomplete and Uncertain Information

James K. Aragones
Piero P. Bonissone
Jonathan Stillman

Artificial Intelligence Program
General Electric Company
Corporate Research and Development
P.O. Box 8
Schenectady, NY 12301

9.1 Introduction and Motivation

PRIMO (Plausible ReasonIng MOdule) is a reasoning system which integrates the theories of plausible reasoning (based on monotonic rules with degrees of uncertainty) and defeasible reasoning (based on default values supported by non-monotonic rules). The PRIMO system consists of a representation language which includes declarative specifications of uncertainty and default knowledge, reasoning algorithms, and an application development environment.

In this paper we review the theoretical foundations of PRIMO (see [BCGS90, BGD87]) and discuss our progress in PRIMO's implementation.

9.2 Uncertainty

The uncertainty representation used in PRIMO is based on the semantics of many-valued logics. PRIMO, like its predecessor RUM [BGD87], uses a combination of fuzzy logic and interval logic to represent and reason about uncertainty. This approach has been successfully demonstrated in two DARPA applications, the Pilot's Associate and Submarine Operational Automation System programs.

PRIMO handles uncertain information by qualifying each possible value assignment to any given propositional variable with an uncertainty interval. The interval's lower bound represents the minimal degree of confirmation for the value assignment. The upper bound represents the degree to which the evidence failed to refute the value assignment. The interval's width represents the amount of ignorance attached to the value assignment. The uncertainty intervals are propagated

and aggregated by Triangular-norm-based uncertainty calculi (see [Bon87, SS63]). The uncertainty interval constrains intervals of subsequent, dependent values.

9.3 Incompleteness

PRIMO handles incomplete information by evaluating non-monotonic justified (NMJ) rules. These rules are used to express the knowledge engineer's preference in cases of total or partial ignorance regarding the value assignment of a given propositional variable. The NMJ rules are used when there is no plausible evidence (to a given numerical threshold of belief or certainty) to infer that a given value assignment is either true or false. The conclusions of NMJ rules can be retracted by the belief revision system, when enough plausible evidence is available.

PRIMO uses the numerical certainty values generated by plausible reasoning techniques to quantitatively distinguish the admissible extensions generated by defeasible reasoning techniques. The method selects a *maximally consistent extension* (see [BCGS90]) given all currently available information.

For efficiency considerations some restrictions are placed on the language in which one can express PRIMO rules. The monotonic rules are non-cyclic Horn clauses, and are maintained by a linear belief revision algorithm operating on a rule graph. The NMJ rules can have cycles, but cannot have disjunctions in their conclusions.

By identifying sets of NMJ rules as strongly connected components (SCC's), we can decompose the rule graph into a directed acyclic graph (DAG) of nodes, some of which are SCCs with several input edges and output edges. PRIMO contains algorithms to efficiently propagate uncertain and incomplete information through these structures at run time. Treating the SCCs independently can result in a significant performance improvement over processing the entire graph. However, this heuristic may result in loss of correctness in the worst case. These algorithms require finding satisfying assignments for nodes in each SCC, and are thus NP-hard in the unrestricted case. We can achieve tractability by restricting the size and complexity of the SCC's, precomputing their structural information, and using run-time evaluated certainty measures to select the most likely extension.

9.4 Implementation

PRIMO has been developed using the NewFlavors object oriented programming language. Most internal PRIMO data types have been represented as NewFlavors objects, e.g., knowledge bases (KBs), rules, and uncertainty intervals. Most of the procedural algorithms have been pushed into the objects, making the system much simpler to develop.

9.4.1 Layers of Abstraction

Internally, PRIMO has three levels of abstraction (see Figure 9.1 below. The first layer, the knowledge base layer, corresponds to the first order predicate calculus. This is the level at which the knowledge engineer writes meta-rules (rules that may contain variables and are assumed to be universally quantified) and the overall design of the KB may be viewed through the rule-class hierarchy.

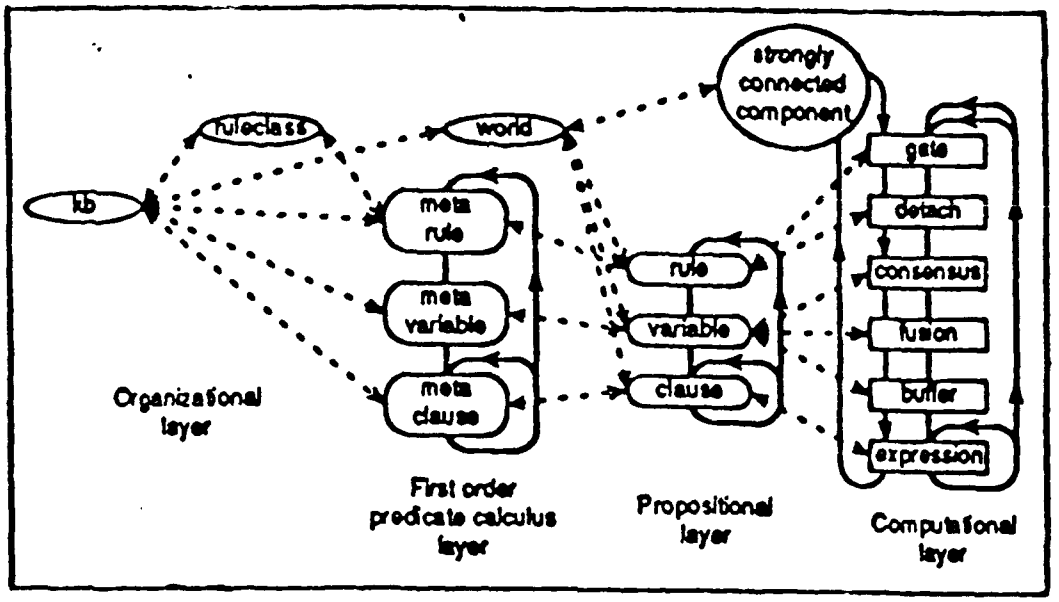


Figure 9.1: PRIMO's three levels of abstraction.

The next layer, the instantiated world layer, is propositional. At this level of abstraction, the meta-rules have been selectively instantiated with ground items. At this point it is possible to maintain values augmented with uncertainty intervals efficiently. In practice, this layer is used for organizational and debugging purposes.

The computational layer is where uncertainty intervals are propagated, user defined predicates are evaluated and SCCs are solved. This layer emphasizes speed of computation at the expense of modifiability, and is used in fielded applications.

9.4.2 Extensible Design

There is an interesting side benefit of using an object oriented strategy for PRIMO; there are only a small number of operations that are performed differently by each node type. Not only does this allow us to eliminate much duplication of code, but it allows us to make quick global changes to the internal operation of the reasoning algorithms, which are distributed throughout the objects. Thus all that

is necessary to forward or backward chain is to write a graph traversal function that generates nodes in the correct order.

9.4.3 Efficiency

Many techniques have been used to improve the performance of PRIMO, especially when operating on monotonic rules. The two methods are used to effectively prune the rule graph before chaining occurs. The most commonly used method is the rule context, a pre-condition that is tested before a rule is fired to ensure rule applicability in the current environment. The second limits rule chaining operations to a specified set of rule-classes, ignoring other rules.

Other techniques have been used to improve the efficiency of the propagation of values through the rule graph. For example, it is possible that two distinct rules may share a portion of their premise clauses. During rule compilation in PRIMO, these portions are recognized as being identical, so the value is only computed once and is shared by both rules. Also, caches are used throughout the intermediate computations for uncertainty intervals. When an interval changes for an input value, all dependent nodes in the graph are signaled that their values are outdated, but it is only when an up-to-date value is required that the rules re-fire.

In a developer's version of PRIMO, there is a large amount of processor and memory overhead required for graphical development tools (described in section 5.2) which would not usually be present in a deployed system. Much of the time spent in the current version of PRIMO is used in creating and destroying objects that are only needed for display. Fuzzy numbers, for instance, are objects which are discarded and garbage collected almost immediately, but they are currently kept as persistent objects for display purposes. By intelligent allocation and deallocation of objects and eliminating objects simply retained for display purposes, we are able to speed computation considerably.

9.5 Knowledge Engineering

9.5.1 Knowledge Base Development

Using plausible reasoning, KBs can be designed using iterative refinement. Knowledge engineering begins by writing a set of simple rules. Later, refinements are made by adding new rules to further constrain the original beliefs. The program's flow of control is usually unimportant to the knowledge engineer; the important features of the KB are the relations between values and the way uncertainty is aggregated.

Rule-class hierarchies are used to organize KB development. In methodology similar to top-down programming, rule-class modules are developed separately. The modules are tested in sets by limiting rule firing to each set. Larger sets are tested similarly until the entire KB is verified.

9.5.2 Development Tools

PRIMO's development environment consists of several graphical displays. One can view uncertainty measures, rule graphs, and hierarchical organization. Many options are available for each of these formats to limit the amount of information displayed at any one time.

In most cases, uncertainty intervals are displayed as sliding bars, the left hand side representing the level of support and the right hand side representing the level of refutation; the width of the bar represents the current level of ignorance (see Figure 9.2).

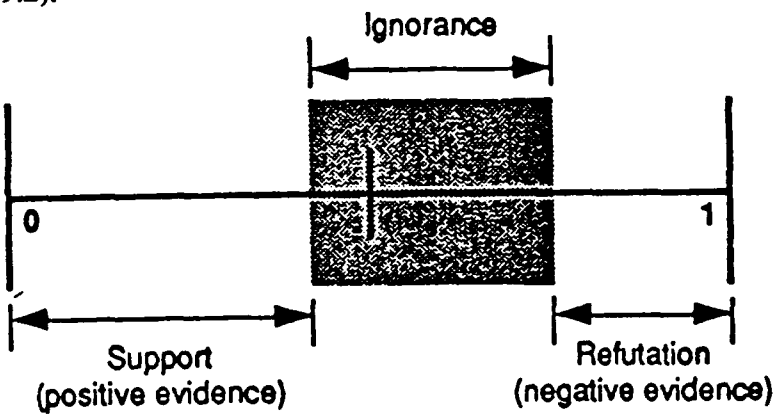


Figure 9.2: A typical uncertainty interval.

PRIMO can also draw complete rule graph layouts to view rule interactions, rule-value interactions and value interactions. It may be important to inspect the contributing factors for a value or the contributions made by a rule. This can be viewed by selecting a node to serve as the root for graphical display (see Figure 9.3).

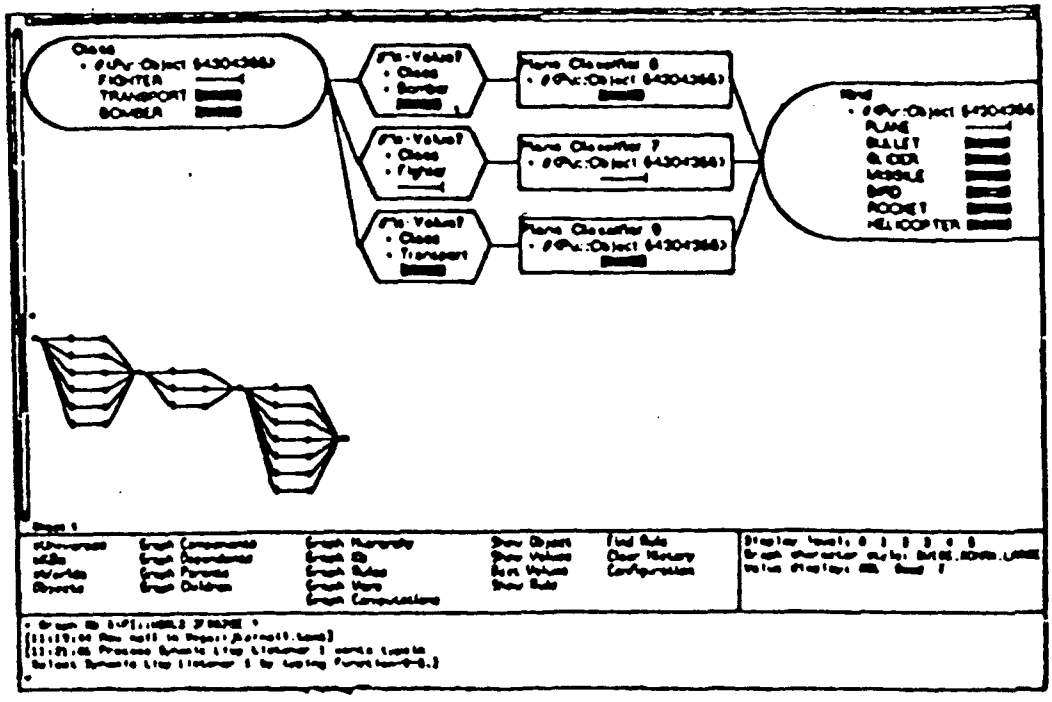


Figure 9.3: A PRIMO rule graph.

Rule-class hierarchies can also be displayed. It is possible to show part of a rule-class hierarchy or show all the rules within a rule-class (see Figure 9.4).

9.6 Conclusions

We have described the theory and an implementation of PRIMO, an approach to integrating plausible and defeasible reasoning. Our current challenge is to test these ideas in a number of applications where uncertainty and incompleteness abound.

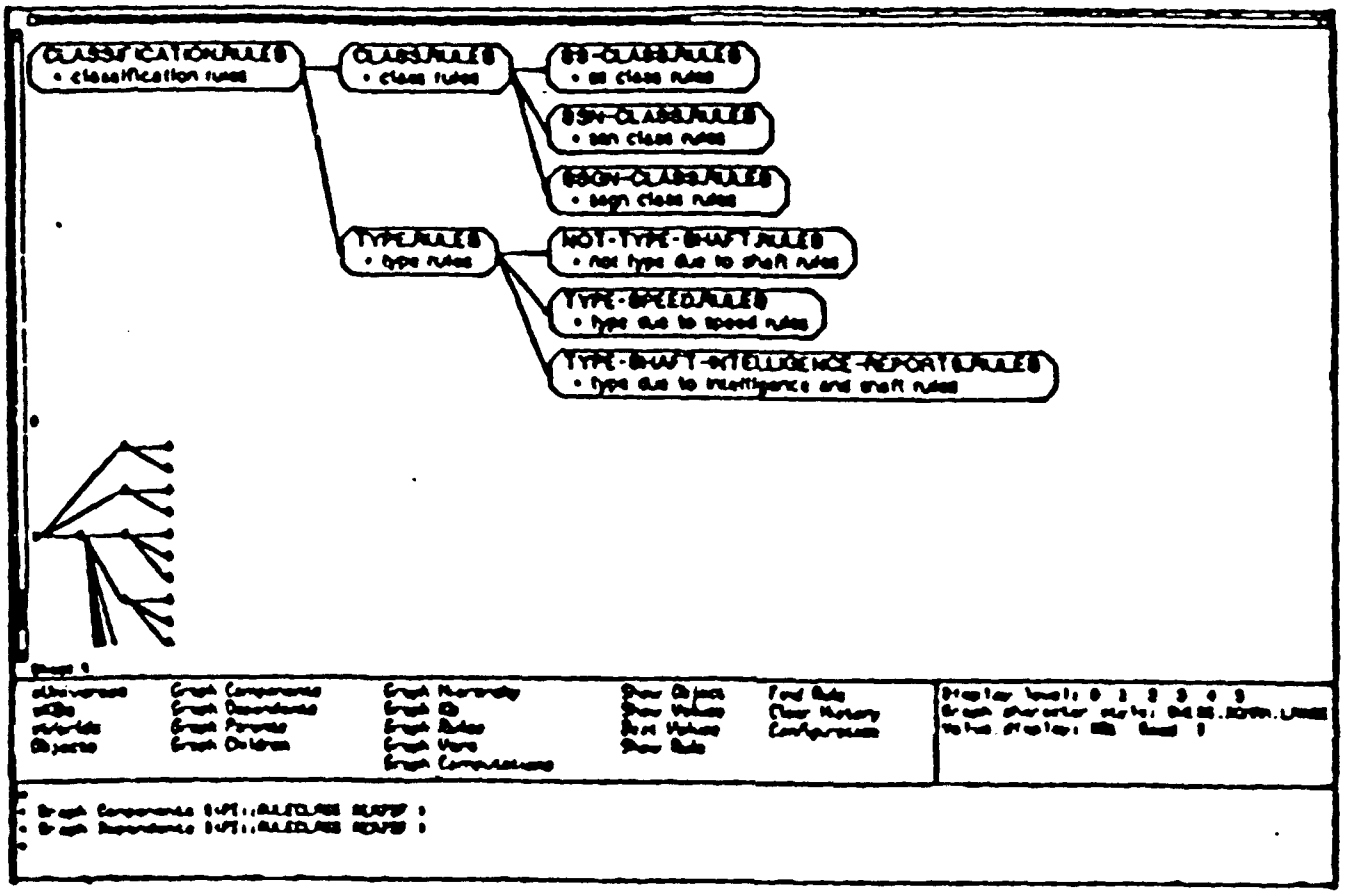


Figure 9.4: A PRIMO rule-class hierarchy.

Bibliography

[BCGS90] Bonissone, P., Cyrluk, D., Goodwin, J., and Stillman, J., August 1989, "Uncertainty and Incompleteness: Breaking the Symmetry of Defeasible Reasoning," in *Proceedings Fifth AAAI Workshop on Uncertainty in Artificial Intelligence*. Windsor, Ontario, Canada.

[BGD87] Bonissone, P., Gans, S. and Decker, K., August 1987, "RUM: A Layered Architecture for Reasoning with Uncertainty," in *Proceedings 10th International Joint Conference on Artificial Intelligence*, Milan, Italy.

[Bon87] Bonissone, P., January 1987, "Summarizing and Propagating Uncertain Information with Triangular Norms," *International Journal of Approximate Reasoning*, 1(1):71–101.

[SS63] Schweizer, B., and Sklar, A., 1963, "Associative Functions and Abstract Semi-Groups," *Publicationes Mathematicae Debrecen*, 10:69–81.

10. PRIMO: User's Guide

Artificial Intelligence Program
General Electric Corporate Research and Development
Schenectady, New York 12301

10.1 Introduction

This chapter serves as an introduction and user's guide for PRIMO. PRIMO (Plausible ReasonIng MOdule) is a reasoning system which integrates the theories of plausible reasoning (based on monotonic rules with degrees of uncertainty) and defeasible reasoning (based on default values supported by nonmonotonic rules). The PRIMO system consists of a representation language (including declarative specifications of uncertainty and default knowledge), reasoning algorithms, and application development tools.

PRIMO is the successor to the Reasoning with Uncertainty Module (RUM), a GE proprietary tool which encapsulated some of the early theory developed before PRIMO. PRIMO itself has been developed using Common Lisp and Symbolics Genera 7.2 Flavors on the Symbolics Lisp Machine.

A complete description of the functions, variables, and macros described in this guide can be found in the *PRIMO Reference Manual*. By convention, actual code (including references to PRIMO system functions and variables) appears in this typeface: `(value buffer)`. Newly-introduced terms and names that stand for other pieces of code (metavariables) appear in *italics*. The names of function keys and other input that you supply to PRIMO appear in this typeface: Function-Help.

10.2 Basic concepts

PRIMO provides facilities for reasoning with uncertain and incomplete reasoning. The uncertainty representation used in PRIMO is based on the semantics of many-valued logics—PRIMO, like its predecessor RUM, uses a combination of fuzzy logic and interval logic to represent and reason about uncertainty. This approach has been successfully demonstrated in two DARPA applications, the Pilot's Associate program and the Submarine Operational Automation System program. PRIMO deals with incomplete information by supporting non-monotonic justified (NMJ) rules, which are used to express the knowledge engineer's preference in cases of total or partial ignorance regarding the value assignment of a given propositional variable.

This section describes the basic concepts behind PRIMO's representation structure, uncertainty propagation algorithms, and inferencing mechanisms.

10.2.1 Reasoning with inference rules

PRIMO represents knowledge about objects and relationships between objects in the form of *inference rules*, which capture the deduction of new facts, or *conclusions*, from sets of given facts, or *premises*. A PRIMO rule is a deductive statement of the form

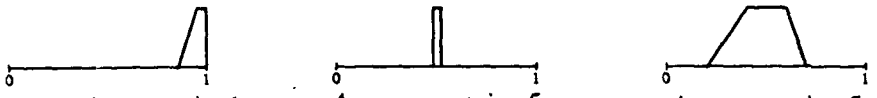
> Given *context*,
> if *premises*
> then conclude *consequences*.

The *context* clause identifies one or more preconditions which must be met before the rule can be applied to the current data. This provides an efficient screening mechanism for the inferencing process, focusing it on a small subset of the entire knowledge base. For instance, one set of rules might be used for determining the intent of friendly agents, and another set used for unknown or hostile agents.

The *premise* clauses are logical expressions, expressed as predicates on attribute values of objects in the current world model. If all of these clauses are satisfied, then the *consequences* are activated. These represent the assignment of values to other object attributes in the world model. Given other rules which then test these consequence attributes in their premises, PRIMO's reasoning processes can construct a series of logical inference chains which support the inferred conclusions.

10.2.2 Representing uncertainty

The basic unit of uncertainty in PRIMO is the *fuzzy* number. In contrast to normal logic, where the truth of a propositional value is either "0" or "1", a fuzzy number can be used to represent truth values across the full range of a particular truth space (which in PRIMO is defined from 0 through 1000) by defining a distribution over the interval. The degree of "truth" associated with the fuzzy number is indicated by the location of the distribution function on the truth interval; the uncertainty associated with this measure is

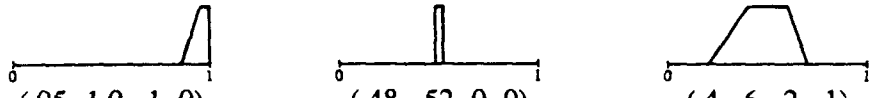


A somewhat certain 1 A very certain .5 An uncertain .5

Figure 10.1: Fuzzy Numbers

indicated by the distribution's width. Figure 10.1 show some typical fuzzy distributions, and their associated interpretations.

For efficiency reasons, the distribution function of a fuzzy number is parametrically characterized by a 4-tuple (a, b, α, β). The first two parameters indicate the interval in which the membership value is 1.0; the third and fourth parameters indicate the left and right width of the distribution, with the membership function varying linearly down to zero between a and $a-\alpha$, and between b and $b+\beta$. Thus, absolute truth is represented by $(1000, 1000, 0, 0)$, absolute falsehood by $(0, 0, 0, 0)$, total ignorance by $(0, 1000, 0, 0)$, and a crisp point x by $(x, x, 0, 0)$. Figure 10.2 shows the same fuzzy distributions previously show in Figure 10.1, but includes their characteristic parameters.



(.95, 1.0, .1, 0) (.48, .52, 0, 0) (.4, .6, .2, .1)

Figure 10.2: Parameterized Fuzzy Numbers

Since it's difficult to obtain precise and consistent numerical certainty values using this notation alone, in PRIMO you can use a number of fuzzy constants called *linguistic terms*. For instance, the constant `*maybe*` represents the fuzzy number $(400, 600, 100, 100)$. The constant `*unlikely*` stands for the fuzzy number $(10, 250, 10, 10)$. Linguistic terms are grouped into a number of *term sets*, which determine the granularity of the measure of certainty that your knowledge base can support. Thus, with the L_7 term set, you could use the following seven terms:

Linguistic Term	Fuzzy Value
`*impossible*`	(0, 0, 0, 0)
`*not-likely*`	(50, 150, 30, 30)
`*small-chance*`	(220, 360, 50, 60)
`*it-may*`	(410, 580, 90, 70)
`*meaningful-chance*`	(630, 800, 50, 60)
`*nearly-certain*`	(830, 960, 70, 30)
`*certain*`	(1000, 1000, 0, 0)

Four other term sets provide greater and lesser degrees of granularity as needed.

In PRIMO, each value assignment to a variable is qualified with two fuzzy numbers, indicating an uncertainty *interval*. The lower fuzzy number represents the minimal degree of confirmation for the value assignment, i.e. how much *positive* evidence there is. The upper fuzzy number represents the degree to which the evidence failed to refute the value assignment, i.e. how little *negative* evidence there is. Like the fuzzy numbers themselves, the distance between the bounds of an interval represents the amount of ignorance attached to the value assignment, and the positions of the lower and upper

bounds indicate the minimum and maximum degree of certainty, respectively, that the value assignment is true. Note that an interval with a lower bound greater than the upper bound indicates *conflict*, with the interpretation that there is both positive and negative evidence with greater than 0.5 certainty.

You may be confused by the difference between fuzzy numbers and the intervals which they define. Both define an interval along the truth spectrum, both are characterized by a degree of certainty associated with the position of the interval on the spectrum, and both have an additional degree of ignorance associated with the interval's width. It might help to think of the interval as sort of a "superfuzzy", whose lower and upper bounds themselves have a degree of fuzziness. Since the widths of both fuzzy numbers are subsumed within the interval, PRIMO is really only using the fuzzy numbers for their position along the truth spectrum. As we'll explain in the next section, however, by explicitly maintaining the lower and upper bounds as fuzzy numbers, we can use a number of useful methods to combine and propagate certainty intervals.

10.2.3 Combining uncertainty measures

Uncertainty intervals are combined and propagated by functions based on special categories of fuzzy operators called Triangular norms, or T-norms. These conjunction and disjunction functions are used to evaluate the satisfaction of rule premises, to propagate uncertainty through rule chaining, and to consolidate the same conclusion derived from *different* rules.

A *T-norm* function $T(x, y)$ aggregates the degree of certainty of two clauses in the same premise. T-norms perform an intersection operation, and at the boundary conditions of 0 and 1 they are equivalent to the logical "AND" operator. A *T-conorm* function $S(x, y)$ aggregates the degree of certainty of the (same) conclusions derived from two rules. These functions perform a union operation, and at the boundary conditions they are equivalent to the logical "OR" operator. Finally, a *negation* function $N(x)$, corresponding to the logical "NOT" operator, is used to group different T-norm and T-conorm functions into pairs, based on DeMorgan's Law, i.e.

$$S(a, b) = N(T(N(a), N(b)))$$

and vice versa.

Using $N(x) = 1 - x$ as the negation operator, PRIMO provides the following five uncertainty calculi:

$T_1(x, y) = \max(0, x + y - 1)$	$S_1(x, y) = \min(1, x + y)$
$T_a(x, y) = \max(0, (\sqrt{x} + \sqrt{y} - 1)^2)$	$S_a(x, y) = 1 - \max(0, (\sqrt{1-x} + \sqrt{1-y} - 1)^2)$
$T_2(x, y) = xy$	$S_2(x, y) = x + y - xy$
$T_b(x, y) = \max(0, \frac{1}{\frac{1}{x} + \frac{1}{y} - 1})$	$S_b(x, y) = 1 - \max(0, \frac{1}{\frac{1}{1-x} + \frac{1}{1-y} - 1})$
$T_3(x, y) = \min(x, y)$	$S_3(x, y) = \max(x, y)$

(see Appendix 10.13 for a discussion on the theoretical basis behind these calculi).

For each calculus, three operations are defined in PRIMO: premise evaluation, conclusion detachment, and source consensus. The *premise evaluation* operation determines the degree to which all the clauses in the rule premise have been satisfied by the input variable values, aggregating the certainty intervals from each clause. If b_i and B_i indicate the lower and upper bounds of the certainty of clauses i in the premise of a given rule, for m clauses, then the combined premise certainty interval $[b, B]$ is

$$[b, B] = [T(b_1, b_2, \ldots, b_m), T(B_1, B_2, \ldots, B_m)]$$

where the T-Norm function $T \in \{T_1, T_a, T_2, T_b, T_3\}$.

The *conclusion detachment* operation indicates the certainty with which the conclusion of a rule can be asserted, given the rule's strength and the aggregated uncertainty of its premise. If s and n are the degree of sufficiency and necessity, respectively, of the rule, and $[b, B]$ is the computed premise certainty interval, as described above, then the certainty interval $[c, C]$ of rule's conclusion is

$$[c, C] = D(s, n, [b, B]) = [T(s, b), N(T(n, N(B)))]$$

where the detachment function $D \in \{D_1, D_a, D_2, D_b, D_3\}$ is defined using the related T-norm function and $N(x) = 1 - x$. The sufficiency and necessity indicate the amount of certainty with which the rule premise implies its conclusion and vice versa. The sufficiency is used with *modus ponens* to provide a lower bound of the conclusion. The necessity is used with *modus tollens* to obtain a lower bound for the complement of the conclusion (which can be transformed into an upper bound for the conclusion itself).

Finally, the *source consensus* operation reflects the fusion of the certainty intervals of the same evidence provided by different sources. If the evidence is an observed fact, fusion occurs before the evidence is used as an input in the deduction process. If the evidence was inferred using two or more rule instances, fusion occurs after the evidence has been aggregated by each group of deductive paths. One type of source consensus operator is the *intersect* function, defined as

$$[d, D] = [\max(c_1, c_2, \ldots, c_n), \min(C_1, C_2, \ldots, C_n)]$$

where c_j and C_j indicate the lower and upper bounds of the certainty of source j, for n different sources contributing a particular value for a variable, and $[d, D]$ is the resulting fused certainty of that variable. Note that if there is inconsistency among some of the sources, the resulting certainty intervals will be disjoint, thus introducing a conflict in the aggregated result. The *dempster-shafer* fusion operator eliminates this by normalizing the intervals before aggregating them.

Thus, there are three different places where you need to specify which calculus to use:

1. For each premise and context clause, if there is more than one predicate you must specify the T-norm with which the predicate results are combined.

2. For each rule, you must specify the detachment operator with which the conclusion detachment will be computed (using the rule's sufficiency, necessity, and premise certainty).

3. Finally, for each rule, you must indicate the consensus operator with which the conclusion aggregation will be computed. This assignment is either *intersect* or *dempster-shafer*.

In assigning premise evaluation and detachment operators, the functions you select will be based on your attitude toward risk for each rule. The ordering of the T-norms spans the range from a conservative attitude (T_1) to a non-conservative one (T_3). From the definition of the calculi operations, T_1 will generate the smallest premise evaluation and the weakest conclusion detachment (i.e., the widest uncertainty interval attached to the rule's conclusion). Higher T-norms will exhibit less drastic behaviors and will produce nested intervals with their detachment operations. T_3 will generate the largest premise evaluation and the strongest conclusion detachment (the smallest certainty interval).

10.2.4 Handling incompleteness

PRIMO handles incomplete information by evaluating non-monotonic justified (NMJ) rules. These rules are used to express the knowledge engineer's preference in cases of total or partial ignorance regarding the value assignment of a given propositional variable. The NMJ rules are used when there is no plausible evidence (with a given numerical threshold of belief or certainty) to infer that a given value assignment is either true or false. The conclusions of NMJ rules may be retracted by the belief revision system when enough plausible evidence becomes available.

PRIMO uses the numerical certainty values generated by plausible reasoning techniques to quantitatively distinguish the admissible extensions generated by defeasible reasoning techniques. The method selects a maximally consistent extension given all currently available information.

For efficiency considerations, some restrictions are placed on PRIMO rules. The monotonic rules are non-cyclic Horn clauses, and are maintained by a linear belief revision algorithm operating on a rule graph. The NMJ rules can have cycles, but cannot have disjunctions in their conclusions.

By identifying sets of NMJ rules as strongly connected components (SCC's), PRIMO decomposes the rule graph into a directed acyclic graph (DAG) of nodes, some of which are SCCs with several input edges and output edges. PRIMO contains algorithms to efficiently propagate uncertain and incomplete information through these structures at run time. Treating the SCCs independently can result in a significant performance improvement over processing the entire graph, although this heuristic may result in loss of correctness in the worst case. The propagation algorithms require finding satisfying assignments for nodes in each SCC, and are thus NP-hard in the unrestricted case. PRIMO attempts to maximize tractability by restricting the size and complexity of the SCC's, precomputing their structural information, and using run-time evaluated certainty measures to select the most likely extension.

10.3 Implementation

PRIMO was developed using the Flavors object-oriented programming language on the Symbolics Lisp Machine (Genera 7.2). Most internal PRIMO data types are represented as Flavors objects, for example, knowledge bases (KBs), rules, and uncertainty intervals.

This section presents a brief overview of those parts of the PRIMO implementation which you should be familiar with.

10.3.1 Abstraction layers

PRIMO has three levels of abstraction. The first layer, the *knowledge base* layer, corresponds to the first order predicate calculus. This is the level at which you write meta-rules (rules that may contain variables and are assumed to be universally quantified). These rules can be organized and grouped into hierarchical ruleclasses.

The next layer, the *instantiated world* layer, is propositional. At this level, the meta-rules have been selectively replaced with ground items representing instantiated rules, predicates, and object instance variables within a particular *world*.

The instantiated nodes are further expanded in the *computational* layer, where uncertainty intervals are propagated, user defined predicates are evaluated, and strongly-connected components are resolved. Each node in this layer represents and implements a computational step in the inferencing process, allowing values to be computed and propagated very efficiently.

10.3.2 Design extensions

Since PRIMO was developed using object-oriented design techniques, there are only a small number of operations that need to be performed differently by each node type. This allows us to reduce duplicated code, and to make quick global changes to the internal operation of the reasoning algorithms, which are distributed throughout the objects. For instance, each object type in PRIMO supports a `free` method, which performs the appropriate resource deallocation and cleanup when an object instance is deleted. As another example, all that is required to implement a new rule chaining strategy is to write a graph traversal function that generates nodes in the correct order, calling an existing `compute` method for each node.

10.3.3 Efficient inferencing mechanisms

Several techniques have been used to improve the performance of PRIMO, especially when operating on monotonic rules. Two methods are used to effectively prune the rule graph before chaining occurs. The most commonly used method is the rule context, a pre-condition that is tested before a rule is fired to ensure rule applicability in the current

environment. The second limits rule chaining operations to a specified set of ruleclasses, ignoring other rules.

Other techniques have been used to efficiently propagate values through the rule graph. For example, two distinct rules may share a portion of their premise clauses. During rule compilation in PRIMO, these portions are recognized as identical, so the value is only computed once and is shared by both rules. Also, caches are used throughout the intermediate computations for uncertainty intervals. When an interval changes for an input value, all dependent nodes in the graph are signaled that their values are outdated, but it is only when an up-to-date value is required that the rules re-fire.

In the developer's version of PRIMO, there is a large amount of processor and memory overhead required for graphical development tools which would not usually be present in a deployed system. Much of the computation time is used in creating and destroying objects that are only needed for display. Fuzzy numbers, for instance, are objects which are discarded and garbage collected almost immediately, but they are currently kept as persistent objects for display purposes. By intelligent allocation and deallocation of objects and eliminating objects used for display purposes, we can speed up computation considerably in a deployment system.

10.4 Using Flavors

Flavors is an extension to Symbolics Common Lisp that supports object-oriented programming. It is a powerful and flexible tool for programming in a modular style. The basic concepts of Flavors are simple to understand and easy to use. On the other hand, Flavors is a complex system that offers many advanced options and programming practices.

This section is included to provide a brief overview of the basic concepts of Flavors (for more information, refer to the section titled "Flavors" in the *Symbolics Common Lisp—Language Concepts* manual). Subsequent sections will discuss the specific use of Flavors in your PRIMO application.

10.4.1 Basic Flavors concepts

Most PRIMO applications are organized around *objects*, which model both real-world things, such as aircraft and submarines, and conceptual entities, such as doctrines and reports. Each object has some *state*, or set of persistent attributes, and a number of operations that can be performed on it. Thus, an object-oriented program consists of a set of objects and a set of operations on those objects.

The Flavors facility enables you to define a new type of data structure that is similar to a KEE's unit or frame. The newly-defined data structure is a convenient, concise, and high-level way to represent an object. Using Flavors terminology, an object-oriented PRIMO application is built around:

Flavors — Each distinct kind of object is represented by a *flavor*, which acts as a template for all objects of that kind. The flavor object defines the inherent structure of its objects.

Flavor instances — Each object is implemented as an *instance* of a particular flavor. The term *object* is used interchangeably with *instance*.

Instance variables — Each flavor specifies a set of state variables for objects of that flavor. These are called *instance variables*. PRIMO extends Flavors by associating certainty intervals with instance variable values, and by providing mechanisms to update these values and their certainty intervals based on rule inferences.

Generic functions — The operations that are performed on objects are known as *generic functions*. Unlike ordinary functions, generic functions may behave a certain way for objects of one flavor, and behave in another way for objects of another flavor.

Methods — The code that performs a generic function on instances of a certain flavor is called a *method*. Typically, one generic function has several methods defined for it, and Flavors chooses which one to use by the flavor of the first argument.

Often a flavor is defined by combining several other flavors, called its *components*. The new flavor *inherits* instance variables and methods from its components, including those which they in turn inherited from their components. Thus, if two types of objects have structure or behavior in common, they can inherit it from the same flavor, reducing duplicated code and increasing extensibility and modularity.

10.4.2 Representing objects

Assume your PRIMO application is dealing with aircraft. You must first determine a way to represent aircraft. If the important things to know about an aircraft are its name, class, type, you can represent aircraft as follows:

```
(defflavor aircraft
(name class type)
        ()      ;no component flavors
  :readable-instance-variables
  :writable-instance-variables
  :initable-instance-variables)
```

The `defflavor` form defines a flavor that represents aircraft. The name of the flavor is `aircraft`. The instance variables are `name`, `class`, and `type`. The definition contains three options, which have the following effects:

`:readable-instance-variables`
: Defines *accessor* functions that enable you to query the object for the value of instance variables. In this case three functions are automatically generated: `aircraft-name`, `aircraft-class`, and `aircraft-type`.

`:writable-instance-variables`
: Enables you to alter the value of instance variables using `setf` and the accessor functions. Note that this option subsumes `:readable-instance-variables`, since writable instance variables are automatically made readable as well.

`:initable-instance-variables`
: Enables you to initialize the value of an instance variable when you make a new instance, using the name of the variable as a keyword.

Once you've defined this flavor, each real-world aircraft in your system can be represented as an instance of `aircraft`. To create new instances, you use the `make-instance` function, as follows:

```
(setf aircraft-1 (make-instance 'aircraft
                                :name "My Aircraft"
                                :class 'fighter
                                :type 'mig-31))
```

10.4.3 Operating on objects

You can query `aircraft-1` for the value of its instance variables by using the accessor functions that were automatically generated as a result of the `:readable-instance-variables` option to `defflavor`. For example:

```
(aircraft-name aircraft-1)
   ==> "My Aircraft"
```

You can also change the value of an instance-variable, using `setf` and the appropriate accessor function:

```
(setf (aircraft-type aircraft-1) 'mig-29)
  ==> MIG-29
```

Finally, you can examine the instance by using `describe`:

```
(describe aircraft-1)
  ==> #<AIRCRAFT 54157652>, an object of flavor AIRCRAFT,
      has instance variable values:
      NAME                "My Aircraft"
      CLASS               FIGHTER
      TYPE                MIG-29
```

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C^3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.